ACNMP: Skill Transfer and Task Extrapolation through Learning from Demonstration and Reinforcement Learning via Representation Sharing

M. Tuluhan Akbulut* Bogazici University, Turkey

Erhan Oztop Ozyegin University, Turkey

M. Yunus Seker Bogazici University, Turkey

Honghu Xue University of Luebeck, Germany

Ahmet E. Tekden Bogazici University, Turkey

Emre Ugur Bogazici University, Turkey

Abstract: To equip robots with dexterous skills, an effective approach is to first transfer the desired skill via Learning from Demonstration (LfD), then let the robot improve it by self-exploration via Reinforcement Learning (RL). In this paper, we propose a novel LfD+RL framework, namely Adaptive Conditional Neural Movement Primitives (ACNMP), that allows efficient policy improvement in novel environments and effective skill transfer between different agents. This is achieved through exploiting the latent representation learned by the underlying Conditional Neural Process (CNP) model, and simultaneous training of the model with supervised learning (SL) for acquiring the demonstrated trajectories and via RL for new trajectory discovery. Through simulation experiments, we show that (i) ACNMP enables the system to extrapolate to situations where pure LfD fails; (ii) Simultaneous training of the system through SL and RL preserves the shape of demonstrations while adapting to novel situations due to the shared representations used by both learners; (iii) ACNMP enables order-of-magnitude sample-efficient RL in extrapolation of reaching tasks compared to the existing approaches; (iv) ACNMPs can be used to implement skill transfer between robots having different morphology, with competitive learning speeds and importantly with less number of assumptions compared to the state-of-the-art approaches. Finally, we show the real-world suitability of ACNMPs through real robot experiments that involve obstacle avoidance, pick and place and pouring actions.

Keywords: Learning from Demonstration, Reinforcement Learning, Deep Learning, Representation Learning

1 Introduction

To equip robots with dexterous skills, an effective approach is to first transfer an approximate version of the desired skill, then let the robot improve it by self-exploration. For the initial transfer of the skill, learning from demonstration (LfD) is a natural choice, where the robot is provided with a set of expert movement demonstrations, by which it learns and reproduces the demonstrations [1]. If the demonstrations are obtained in the intrinsic coordinates of the robot, a simple playback controller, or a neural network capturing the state to action mapping can synthesize the initial skill (e.g. [2, 3]). To improve the quality of the initial skill obtained, and/or to adapt it to novel environments, the movement policy of the robot is typically modified via Reinforcement Learning (RL) [4], where the reward function defined by the designer determines how the initial skill is to be modified (e.g. [5]).

Instead of targeting a single task, recent efforts in LfD+RL approaches focus on developing methods to allow the representation of multiple policies that can be flexibly adapted according to the goals and contexts, while retaining original skills if needed [6]. We believe that this is one of the keys for paving the road for life-long learning. We consider the following three challenges in this context:

*Corresponding author: `tuluhan.akbulut@boun.edu.tr`

[0]Code is available at:`https://mtuluhanakbulut.github.io/ACNMP`

Sample-efficiency. The relations between task parameters and the required policy are often highly non-linear and complex in real-world problems. Consequently, excessive policy updates may be required to reach a policy to make the robot perform in new contexts. Therefore, sample-efficient robust machine learning methods are needed to adapt the initial policies to the novel ones [7].

Avoiding interference of novel goals with the initial demonstrations. The skill demonstrations provided by humans include intrinsic features that are difficult to encode in a reward function. These characteristics might be related to safety, robot and environment constraints, or other user-preferences. Thus, the second challenge is to update the policy towards the fulfillment of novel contexts/goals while maintaining the characteristics of the initial demonstration.

Skill transfer between agents with different embodiments. When the skill to be transferred is not represented in the robot intrinsic coordinates then the skill transfer requires the additional step of converting the demonstration to the robot intrinsic coordinates. This is problematic if the mapping is not well defined or multiple solutions exist, in which case, it is desirable to constraint the mapping by the space of demonstrations. Therefore, an important challenge in transferring skills between agents is to develop mechanisms that enable such transfer automatically without explicitly designing the coordinate between agents.

In this study, we address the aforementioned challenges by: **(i)** Developing an LfD+RL framework that allows the robot to learn complex relations between the task parameters and generated motions in a sample-efficient way. **(ii)** Ensuring the LfD+RL framework to adapt the policy to new environments while reflecting and maintaining skill knowledge captured from the initial demonstrations. **(iii)** Creating a common representation space between robots with different embodiments, which allows automatic skill transfer between them.

To address these challenges, we leverage the recent advances in deep learning, particularly in Conditional Neural Processes[8], and extend a previously formulated LfD framework[9] with a novel RL component. Our proposed LfD+RL framework, namely Adaptive Conditional Neural Movement Primitives (ACNMP) learns the distribution of the input movement trajectories conditioned on the user set task parameters by representing the relations between the task parameters and movement trajectories from a few demonstrations. When the system is queried outside of the range it was trained for, the generated trajectories may fail to achieve the desired goals. This may also happen when the system is placed in a new environment. In these cases, the system starts to adapt its skill representation through simultaneous RL and LfD learning. To be concrete, the system updates its internal parameters via RL guided actions, which is alternated with error-based supervised learning (SL) based on the demonstration set. An ACNMP system employs an encoder-decoder network with a powerful latent space representation (i.e. a CNMP system [9]) for implementing the LfD part. In ACNMP, this network, in a novel way, is also given to implement the policy network for the RL part. The ACNMP architecture not only allows acquiring an ever growing set of movement trajectories but also facilitates skill transfer between two morphologically different robots. To do this, two ACNMP models are trained to develop a common representation in their latent layers by using a proxy skill demonstration that is available to both. Then one robot can automatically learn a novel, possibly more complex, task by observing its execution by its peer.

We conducted simulation experiments to justify the aforementioned features of ACNMP and show its superiority over the existing methods. Moreover, through real robot experiments, we showed the suitability of ACNMP for real-world deployment.

2 Related Work

Learning Movement Primitives from Demonstrations LfD [1] has been extensively used in robotic problems including object grasping and manipulation [10, 11, 12, 13, 14]. Among others, learning methods that are based on dynamic systems [15], statistical modeling [16] and their combination [17, 18] have been popular in recent years. Dynamic Movement Primitives (DMPs) encode the demonstrated trajectory as a set of differential equations, implementing a spring-mass-damper system extended with a non-linear function. Encountered with novel situations, the model parameters of the DMP can be adjusted using RL [19, 20]. While DMPs generate deterministic trajectories, Probabilistic Movement Primitives (ProMPs) [21] can encode a distribution of trajectories and generate stochastic policies. CNMPs [9] can also encode trajectory distributions while learning non-linear relationships. We select CNMPs as our LfD model and we will compare the performance and generalization capabilities of our method (ACNMP) against adaptive ProMPs in this paper.

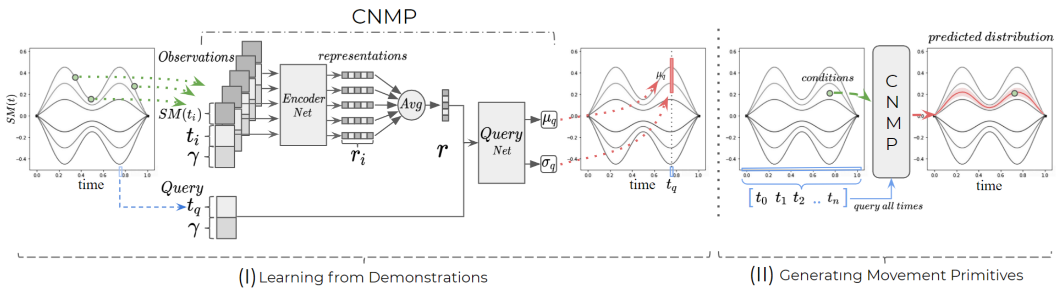

Figure 1: Policy network training and trajectory generation during LfD. See Section 3.A for details.

Adaptation of Primitives Sample efficiency is critical for RL when applied in the real-world, henceforth, LfD is often combined with RL to facilitate sample-efficient learning [7]. These approaches can be viewed as two complementary categories. The first one employ an RL engine at the core and use demonstrations to help RL by providing expert input [22, 23, 24, 25]. For example, in [22], consistency between the demonstrated and self-acquired trajectories is maintained by defining a trade-off cost function that takes into account expert knowledge in integrating the reward obtained from the environment. However, unless the trade-off weighting hyper-parameters are carefully tuned, the agent may get stuck with the performance of the demonstrator or fail to benefit from the demonstration [26]. The second category, which also includes our work, takes an LfD engine as the core learner and improves its performance by extending it with RL. Here, our approach is closely related to [27, 28], where ProMPs are used to encode demonstrations and they adapt ProMPs to new task constraints via RL. In [27], a planar robot arm learns separate ProMPs for pushing objects to different goal points through demonstrations, and extends this skill to novel goal points by exploiting the previously learned ProMP models. The KL divergence is used to stay close to previous parameters of the model, avoiding distorting the shape of the movement [29]. In our study, the same skill with different task parameters (such as goal points) is encoded in a single ACNMP model. Similar to ours, [28] also uses a single model which combines ProMP and Gaussian Processes to encode a parametric skill and condition it with the corresponding task parameters. For adaptation, RL is used to find the relations between the task parameters and the ProMP model parameters by using a trajectory relevance metric. Compared to the above studies, our model does not require explicit optimization using metrics such as additional/weighted loss, relevance or KL-divergence. Instead, ACNMP is trained together with the demonstrated trajectories and the newly explored ones, automatically preserving the old skills while extending the model to the new task parameters thanks to the robust and flexible generated representations. The idea of augmenting the demonstrations in online learning settings is also studied in [30], where only supervised learning is employed.

Transfer learning has also been an important challenge in robotics [31]. The correspondences of states between different agents are established by considering all state pairs [32], by manually designing a common feature space [33], or by aligning states via unsupervised manifold alignment [34]. Gupta et al. [35] aligns the states using expectation-maximization based dynamic time warping, and then found a common feature space using non-linear embedding functions. Different from the previous studies, our method does not require an explicit state matching step. It forms a common feature space that encodes the correspondence of the trajectories rather than single states.

3 Method

As mentioned in the introduction, our system is composed of LfD and RL components. The **LfD component** of our system is composed of an encoder-decoder network, namely CNMP [9]. The encoder layer of the CNMP learns a representation of trajectory points and other variables conditioned on the time. The decoder layer takes the learned representations and outputs the mean and variance of a Gaussian distribution as a function of time. Figure 1.I shows the training procedure of a CNMP using a hypothetical 1D scenario. In each training iteration, a changing number of random sensorimotor time and value pairs, named observation points, are sampled from a random demonstration trajectory. The observation points are passed through the parameter sharing encoder network in order to be transformed into their corresponding latent space representations, which are then merged into one single general representation. The produced general representation is concatenated with the target query time value and passed through the decoder network that produces a mean and a variance, which corresponds to the sensorimotor value distribution of the query time based on the sampled observations. The network is trained end-to-end with stochastic gradient descent

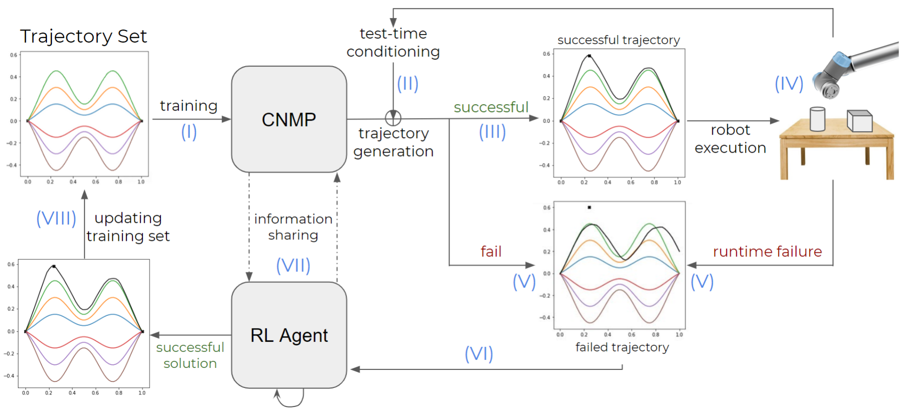

Figure 2: The proposed ACNMP architecture and the learning loop.

algorithm via the loss function: $\mathcal{L}(\theta, \phi) = -\log P(SM(t_q) \mid \mu_q, \text{softplus}(\sigma_q))$ where μ_q and σ_q are the predicted distribution parameters over the queried time-step t_q, and $SM(t_q)$ is the recorded sensorimotor value at the queried time. Conditioning on task parameters (γ) is achieved by concatenating the parameters with the time points. After training, the CNMP can be queried to produce a trajectory that matches a desired set of outputs at desired time-points (Fig. 1.II).

The **RL component** of our system follows a policy gradient RL approach, where the behavior of an RL agent with stochastic policy is governed by the policy $\pi_\theta(a, s) = P(a_t \in A | s_t \in S, \theta)$, where S and A denote state and action spaces, and θ indicates the parameters of a function approximator. We define the policy to depend on context c, which is composed of time and possibly task parameters. Thus we have $\pi_\theta(a, s, c)$ as our policy. As our agent needs to generate continuous actions, we adopt a policy-gradient-based algorithm for learning. Policy gradient algorithms aim to maximize the expected total reward $R(\tau) = \sum_{t=1,..T} R(s_t, a_t, c_t)$ over a state-action trajectory $\tau = \{s_0, a_0, ...s_T, a_T\}$ with respect to the policy parameters θ [36]. In our setting, the update on policy parameters θ becomes:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(s_t, a_t, c_t)(\sum_{t'=t}^{T} R(s_{t'}, a_{t'}, c_{t'}))] \tag{1}$$

Fig. 2 provides the general proposed LfD+RL framework. Given the demonstration trajectories, the base CNMP model is trained (I) following the steps outlined in the first paragraph. In order to achieve the goal with given task parameters, CNMP model is conditioned with the task parameters (II) and generates the required trajectory (III), which is executed by the robot (IV). In case a failure is observed in extrapolation cases (V), RL agent steps in (VI) and updates the CNMP model via policy gradient approach ; while the same model continues being trained using demonstration trajectories (VII). After the RL agent finds the optimal trajectory for the new task , this new trajectory is regarded as a new demonstration and fed back to the LfD system (VIII).

Simultaneous Training with SL and RL: A crucial aspect of our framework is that during the application of RL updates to discover a suitable trajectory, learning based on expert trajectories is simultaneously carried out to preserve the learned knowledge. This procedure ensures the formation of a model that not only produces novel trajectories that satisfy the new constraints but also reproduces previously learned trajectories. In practice, this allows the system to generate novel trajectories that share similar characteristics with the demonstrated ones. We train the neural network using two loss functions for corresponding objectives, 1 and the maximum likely-hood loss function of CNMP to achieve the aforementioned effect. For the inputs (observation and task parameters) coming from demonstrations, supervised learning loss is used to update the network. When the inputs are novel (coming from extrapolation cases), the same network is updated by policy gradient loss. If we manipulate the CNMP loss in equation 1, it can be seen that it is similar to policy gradient loss:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\text{CNMP}_\theta}[(\sum_{t=1}^{T} \nabla_\theta \log_{\text{CNMP}_\theta}(a_{t,\text{demo}}, c_t))] \tag{2}$$

where $R(\tau) = 1$ as the true action ($a_{t,\text{demo}}$) is known. This manipulation can be thought as follows: policy gradient adjusts policy distribution to make the most rewarding action most likely whereas

CNMP adjusts trajectory distribution to make the observed actions most likely. Note that the imbalance in the number of initial demonstrations and the new trajectories does not hurt the target distribution since CNP can represent multimodal distributions [8].

Assimilating New RL Solution into the CNMP: In finding the new solution, RL agent only maximizes the reward while using only conditioning points as observations. However, CNMP is trained by sampling random observations from the trajectory. Therefore, the solution proposed by RL is added to the trajectory dataset and CNMP is trained further as described at the beginning of this section 3. This assimilation step ensures the capability to condition CNMP from any point in the range of previous expert demonstrations and newly found RL solution.

Transfer Learning: To transfer knowledge between different agents, two agents initially learn skills with their own ACNMPs that are trained together for the proxy tasks to find common latent representations from demonstrations. Then, the target agent uses these representations with RL to learn the test task demonstrated by the source agent. Note that the problem of matching actions executed by robots with a different number of joints and/or sampled with different frequencies is naturally handled by the proposed system as whole trajectories are mapped to the same length latent vectors. This mapping is achieved by requiring the latent spaces of two CNMPs to be as close as possible by including a latent space distance term in the loss function:

$$\mathcal{L}(\theta_1, \phi_1, \theta_2, \phi_2) = \mathcal{L}_{\text{CNMP1}}(\theta_1, \phi_1) + \mathcal{L}_{\text{CNMP2}}(\theta_2, \phi_2) + \mathcal{L}_{\text{MSE}}(R_1, R_2) \tag{3}$$

where $\mathcal{L}_{\text{MSE}}$ stands for mean squared error, R_1 denotes representation coming from encoder of first CNMP and R_2 denotes representation coming from encoder of the second CNMP. After the latent space is formed, the knowledge transfer scenario folds out as follows. It is assumed that the first robot knows how to solve the test task; yet, the second robot has no knowledge of the task, and it would need many iterations to learn to solve the task. In this case, by looking at the solution trajectory of the first robot, the latent space representation can be found by the encoder of the first CNMP. Giving this representation to the decoder of the second CNMP results in the interpretation of the first robot's solution in terms of the second robot's joint angles. Afterwards, the RL agent can optimize the trajectory quickly to solve the task.

4 Experimental Results

The performance of our system is evaluated in four simulations and two real robot experiments. For two experiments with which we compare our methods against the source code was not available, so the results pertaining those are obtained with our implementation based on the algorithms (including reward functions) given in Stark et al. [27], Gupta et al. [35]. The implementation details can be found in the Appendix A.

4.1 Extrapolation in 2D Obstacle Avoidance

This section aims to first verify that the original CNMP formulation fails to generalize when conditioned with points outside the learning range and then to evaluate the effectiveness of our proposed method. For this purpose, we simulated an obstacle avoidance task in a 2D world where different trajectories are given as demonstrations that avoid static obstacles. The six demonstrations that start from the same position and end at the same position are shown in Fig. 3(a) with different colors. Given a different via-point (shown with a dot), a trajectory whose shape is similar to the demonstrated ones is expected to be generated. However, conditioned with a dot outside of the training range, LfD model generates a disparate trajectory that does not pass through the conditioned point (Fig. 3(b)). ACNMP adapts network weights using previous demonstrations via supervised learning and new trajectories generated by RL setting the reward as the minus distance between the generated and the target points. After 35 roll-outs, ACNMP generates a trajectory passing through the given purple-dots while preserving the shape similar to initial demonstrations (Fig. 3(d)). Note that, when only RL is applied on the learned LfD model, the system can also generate a trajectory passing through the required point, but without preserving the shape (Fig. 3(c)). We further analyzed our ACNMP system in extreme extrapolation cases and observed that while the constraints are satisfied, the shapes of the trajectories diverge from the demonstrated ones (see Appendix B for details).

Assimilating new constraints into CNMP representations: We investigate how different demonstrations are encoded in the representation layer of the neural network and how this representation changes after ACNMP learning. For this purpose, points from the trajectories are sampled and the corresponding activations in the 2-neuron latent representation layer are visualized in Fig. 4. As

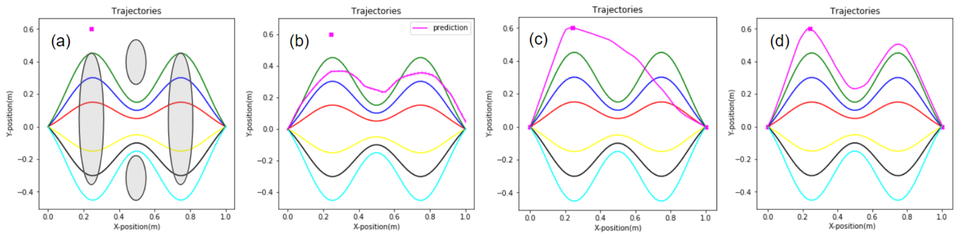

Figure 3: (a) The simulated 2d obstacle avoidance task. The 6 colored trajectories correspond to the demonstrated trajectories, the gray ellipses represent the underlying hypothetical obstacles and the pink red point shows the conditioned point. (b) LfD model fails to extrapolate. (c) Only RL-based LfD trajectory. (d) Simultaneous RL & LfD-based ACNMP.

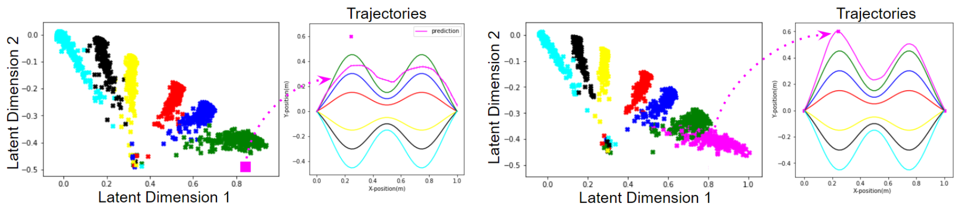

Figure 4: Illustration of the representations for each trajectory in the latent representation space. The left and right parts show latent representation before and after RL solution is assimilated into the model, respectively.

shown, each separate trajectory is represented in a different cluster in the latent space. After RL is applied and the newfound trajectory is integrated into the CNMP, it is represented in a separate cluster in the latent space. Note that, if the same activation were used in the latent space before the RL step, the reproduced trajectory could not satisfy the constraints (as shown in the top-right plot). This analysis shows that our method enables the robust integration of new constraints into the existing representations.

4.2 Adaptation to Novel Environments in Simulation Experiments

We evaluate the extrapolation capabilities of ACNMP in a simulated pushing and obstacle avoidance tasks, and compare the results with the state-of-the-art approaches [27, 28] in two experiments. In the first experiment, we replicated the experimental setup of [27] in CoppeliaSim [37]. In this task, the task of a joint-controlled 3-dof planar robot arm is to push a cylindrical object from a fixed position to one of 10 different target positions. The robot is provided with demonstrations for 9 different target points and is expected to generate a trajectory and adapt if necessary to push the object to the remaining target point (Fig. 5 (a)). Both [27] and us set the reward to minus distance between the generated and target positions of the object. In [27], pushing the cylinder to each target was represented by a single ProMP model, whereas the joint trajectories were conditioned on target position in our model, therefore all skills were encoded into a single ACNMP. We analyzed two different situations, namely interpolation and extrapolation cases, separately and provided the performance of ACNMP in Fig. 5 (b). As shown, ACNMP could accomplish the interpolation tasks already from the beginning and adapt to extrapolation tasks by sampling around 130 roll-outs. In comparison, [27] achieved the task using around 1200 roll-outs where the majority of the cases (80%) were from interpolation. In short, ACNMP does not require further adaptation as in [27] for interpolation cases, and it achieves superior performance especially for extrapolation cases.

Table 1: Comparison summary

	ACNMP	ProMP+RL
# of traj.	130	1200 [27]
# of traj. (single env.)	1.4K	20K [28]
# of traj. (total)	0.7M	10M [28]

In the second experiment, we replicated the 2D obstacle avoidance experiment [38] of Ewerton et al. [28], which optimizes trajectories based on the concept of relevant and generates trajectories using given task parameters via RL. In this experiment, we show that demonstrations for ACNMP do not need to be aligned temporally, please refer to the Appendix C for details. As shown in Fig. 5 (c) the task of the system is to generate obstacle avoidance trajectories that reach given goal points. The task parameters are composed of coordinates of the center of the hole in wall(x_c, y_c) and coordinates of the goal point (x_g, y_g), and RL minimizes the distance to the start point, the distance to the goal point and the signed distance to the walls in both [28] and our ACNMP model (see the Appendix C for details.). In order to create training and test environments, these coor-

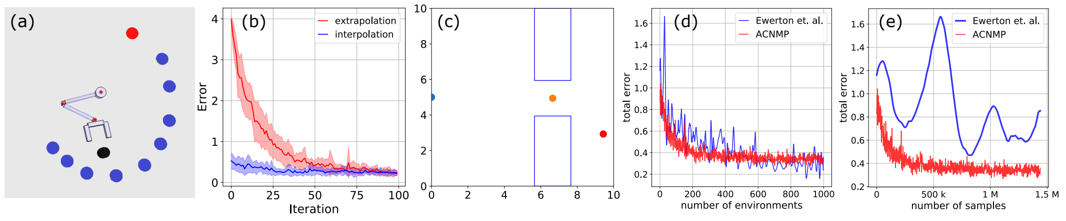

Figure 5: Tasks and performance comparisons for Stark et al. [27] (a-b) and Ewerton et al. [28] (c-e)

dinates are sampled from a uniform distribution from the range of $2 \leq x_c \leq 8$, $1 \leq y_c \leq 9$, $x_c + 1.5 \leq x_g \leq 10$ and $0 \leq y_g \leq 10$. Both models are initialized with 30 random environments. Next, the models use a self-improvement loop via RL to increase the performance gradually. The performance of both methods is evaluated using a distinct test set composed of 1000 environments and figures 5 (d) and 5 (e) provides the change in error with respect to the number of evaluated environments and trajectories, respectively. As shown, while both methods are competitive in terms of the number of environments used, ACNMP outperforms the [28] when the number of sampled trajectories are considered. Because ACNMP can learn the avoidance task for each environment from 1400 new trajectories maximum, their method required 20,000 new trajectories. In summary, the performance achieved by ACNMP with 700K trajectory samples could be reached by their method using 10 Million samples.In the table 1, ACNMP's performance is summarized in comparison with [27] and [28]. For each environment and task, ACNMP requires fewer samples due to model's high representation capabilities and continuous exploitation of learned representations during RL. Note that our model optimizes more parameters as it uses neural networks (CNP [8]) at the core, which has a higher computational requirement.

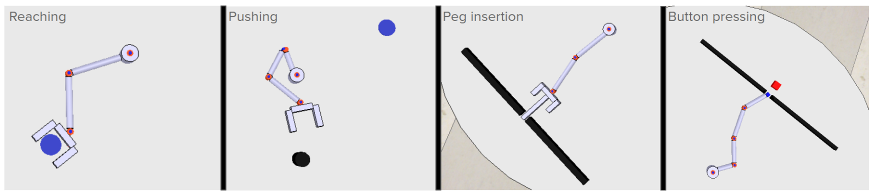

Figure 6: The three training tasks (on the left) and the test task (on the right) for transfer learning.

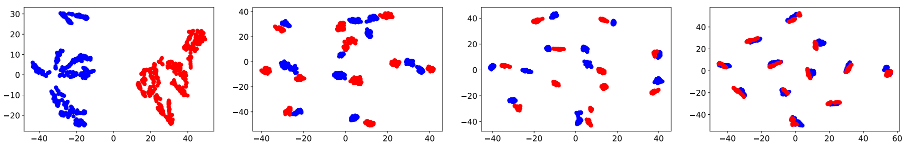

Figure 7: Snapshots of overlaid latent space representations obtained during learning.

4.3 Skill Transfer between Robots with Different Morphologies

This experiment aims to evaluate the transfer learning capability of ACNMP, where two robots with different morphology are provided with the demonstration of a proxy skill. Additionally, demonstrations from a new task is made available only to the first robot. Then, we evaluate how fast the second robot can learn the new task by exploiting the behaviour of the first robot on the new task. For this, two models of the robots are trained for the proxy skill while forcing the latent representation in each robot's network to be similar. We adopted the experiment setup in [35] for transfer learning. The two (3- and 4-dof) planar robot arms are simulated in CoppeliaSim [37] using the PyRep toolkit [39]. Both robots are provided with four different trajectories for reaching, pushing and peg insertion as shown in Fig. 6. First, our system learns these skills and forms similar latent representations in the two robots (see Fig. 7). The 3-dof robot is further provided with new demonstrations for a new task, namely button pressing task, where the robot must pass through a narrow opening and push the blue cube to the position of the red cube (Fig. 6). The latent space activation is transferred to the 4-dof robot, which initially generated a trajectory close to the required one, but could not achieve the task. Via the same sparse reward used in [35], the distance between the two cubes, our ACNMP model was able to achieve 23, 73 and 90 percent success rate in 3, 7 and 11 iterations respectively and reached perfect success rate in 19 iterations. While both methods show similar performances, [35] obtained these results through explicitly learning a pairing of states observed in two robots and

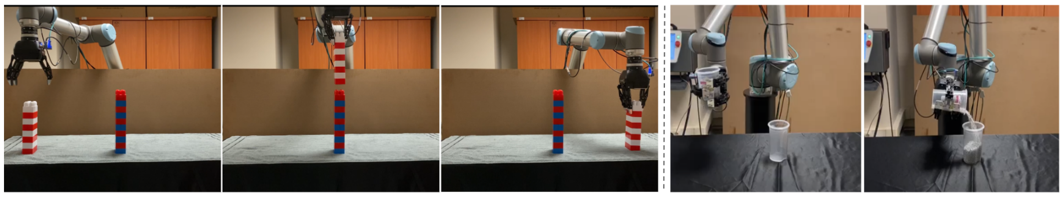
Figure 8: Left: 3 snapshots from the pick & place task. Right: 2 snapshots from the pouring task.

Table 2: Left: Average joint errors in pick & place task. Right: Error change in pouring during RL.

		Joint Errors (deg)					
Method	**Parameters**	**Base**	**Shoulder**	**Elbow**	**Wrist1**	**Wrist2**	**Wrist3**
CNMP	Obj: 8* Obs.: 10*	1.038	2.325	8.879	8.094	0.360	0.830
ACNMP	Obj: 8* Obs.: 10*	0.889	1.589	4.668	3.465	0.845	0.861
CNMP	Obj: 9* Obs: 11*	1.861	5.380	17.978	14.894	0.590	1.386
ACNMP	Obj: 9* Obs: 11*	0.994	0.507	1.780	1.514	0.814	1.007

	Error in Grams			
1-4	591	447	435	175
5-8	511	604	305	42
9-12	417	284	144	194
13-16	367	364	79	177
17-18	133	**27**		

but they required an explicit time-warping mechanism. In our case, importantly, no time-warping and no explicit mechanism that ensures state-to-state match were required; our method learned the correspondences in trajectory level.

4.4 Extrapolation in Real-Robot Adaptation

The first real robot experiment [40] aims to verify the ACNMP in extrapolating to conditions in an object pick and place task that requires obstacle avoidance using a 6-dof UR10 robotic arm mounted with a Robotiq gripper (Fig. 8). LfD-only model failed to generate suitable trajectories in configurations with objects and obstacles higher than the demonstrated ones [9]. Similar to [9], ACNMP was provided with 8 demonstration trajectories for object and obstacle configurations from the range of [2-6] cm and [2-8] cm, respectively. The task parameters are set as the heights of the objects and the reward is set as the sum of distances to the grasp and obstacle avoidance points. ACNMP was able to adapt to novel task parameters (i.e. object and obstacle heights) that are non-linearly related to the required trajectories (Table. 2 left).

Above, the extrapolation capability of the ACNMP is verified when it is conditioned on novel points in its movement trajectory. The aim of the second real robot experiment is to show that ACNMPs can also adapt when conditioned on task parameters that are outside the range of training and using the reward coming from the real-world. The robot is given a cup that contains different amounts of marbles with the aim of pouring the target amount of marbles into another cup (Fig. 8 right). Depending on the weight of the cup and the target pouring amount, the robot is expected to generate wrist joint trajectories with suitable amount of rotation. Therefore, ACNMPs need to learn to generate joint trajectories conditioned with the initial weight of the cup and desired pouring amount. 16 expert joint trajectories were given with initial cup weights in the range of 650-1200 grams and rotation angles in the range of 85-95 degrees. Given the extrapolation parameter, 1400 gram initial cup weight and 1322 grams target pouring amount, ACNMP found an RL solution and assimilated it into the primitive after 18 roll-outs. While the LfD approach showed 600 gram error on this extrapolation task, the error rate of ACNMP dropped to 27 grams after training (Table 2 right). In real robot experiments, we used an explicit controller to drive the robot since ACNMP produces trajectories.

5 Conclusion

In this paper, we proposed a new adaptive movement primitive approach, namely ACNMPs, by integrating LfD and RL to address the challenges of efficient sampling, skill maintenance and skill transfer. ACNMPs have been shown to outperform two other adaptive movement primitive learning approaches in solving extrapolation tasks in terms of adaptation performance and data-efficiency while still retaining the qualitative characteristics. ACNMPs were shown to learn meaningful latent feature representations from the data, thus contributing to a successful generalization. As a step further, we exploited the latent representation to perform domain transfer between robots with different morphologies to solve an unknown task in a few-shot learning manner. Finally, the applicability of the ACNMPs to real-world scenarios was demonstrated by real robot experiments involving obstacle avoidance in reaching and manipulation tasks. In the future, we would like to seek mathematical investigations and theoretical guarantees on shape preservation and assess our method on tasks including rich contacts with real robots.

Acknowledgments

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 731761, IMAGINE. It was partially supported by JST CREST "Cognitive Mirroring" [grant number: JPMJCR16E2], by the International Joint Research Promotion Program, Osaka University and TUBITAK (Scientific and Technological Research Council of Turkey) 2210-A scholarship. We would like to thank Yukie Nagai, Svenja Stark and Elmar Rueckert for their comments and feedback on this work and the reviewers for their suggestions.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Rob. and Auto. Sys.*, 57(5):469–483, 2009.

[2] L. Peternel, T. Petric, E. Oztop, and J. Babic. Teaching robots to cooperate with humans in dynamic manipulation tasks based on human-in-the-loop approach. *Auton Robots*, 36, 2014.

[3] J. Babič, J. G. Hale, and E. Oztop. Human sensorimotor learning for humanoid robot skill synthesis. *Adaptive Behavior*, 19(4):250–263, 2011.

[4] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[5] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, page 0278364913495721, 2013.

[6] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. *Foundations and Trends in Robotics*, 7(1–2):1–179, 2018.

[7] G. Dulac-Arnold, D. Mankowitz, and T. Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[8] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. M. A. Eslami. Conditional neural processes. In *ICML*, 1704-1713 2018.

[9] M. Y. Seker, M. Imre, J. Piater, and E. Ugur. Conditional neural movement primitives. In *Robotics: Science and Systems (RSS)*, 2019.

[10] S. Calinon, P. Evrard, E. Gribovskaya, A. Billard, and A. Kheddar. Learning collaborative manipulation tasks by demonstration using a haptic interface. In *Intl Conf Adv Robotics*, 2009.

[11] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*, 5(02):183–202, 2008.

[12] H. Ben Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters. Generalization of human grasping for multi-fingered robot hands. In *IROS*, pages 2043–2050. IEEE, 2012.

[13] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. In *IROS*, 365–371, 2011.

[14] M. Mühlig, M. Gienger, and J. J. Steil. Interactive imitation learning of object movement skills. *Autonomous Robots*, 32(2):97–114, 2012.

[15] S. Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, page 261. Springer, 2006.

[16] S. Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016.

[17] H. Girgin and E. Ugur. Associative skill memory models. In *IROS*, pages 6043–6048, 2018.

[18] E. Ugur and H. Girgin. Compliant parametric dynamic movement primitives. *Robotica*, 2020.

[19] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *J. Mach. Learn. Res.*, 11:3137–3181, Dec. 2010. ISSN 1532-4435.

[20] A. Colomé and C. Torras. Dimensionality reduction and motion coordination in learning trajectories with dynamic movement primitives. In *IROS*, 2014.

[21] A. Paraschos, C. Daniel, J. Peters, and G. Neumann. Probabilistic movement primitives. In *NIPS*, pages 2616–2624, 2013.

[22] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al. Deep q-learning from demonstrations. In *AAAI*, 2018.

[23] M. Vecerik, O. Sushkov, D. Barker, T. Rothörl, T. Hester, and J. Scholz. A practical approach to insertion with variable socket position using deep reinforcement learning. In *ICRA*, 2019.

[24] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2018.

[25] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess. Reinforcement and imitation learning for diverse visuomotor skills, 2018.

[26] Y. Li, I. Kash, and K. Hofmann. Learning good policies from suboptimal demonstrations. In *The 14th European Workshop on Reinforcement Learning (EWRL 2018)*, 2018.

[27] S. Stark, J. Peters, and E. Rueckert. Experience reuse with probabilistic movement primitives. In *IROS*, 2019.

[28] M. Ewerton, O. Arenz, G. Maeda, D. Koert, Z. Kolev, M. Takahashi, and J. Peters. Learning trajectory distributions for assisted teleoperation and path planning. *Front. Robotics and AI*, 2019.

[29] J. Peters, K. Mulling, and Y. Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[30] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Intl Conf on artificial intelligence and statistics*, 2011.

[31] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.

[32] M. E. Taylor, N. K. Jong, and P. Stone. Transferring instances for model-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 488–505, 2008.

[33] H. B. Ammar and M. E. Taylor. Reinforcement learning transfer via common subspaces. In P. Vrancx, M. Knudson, and M. Grześ, editors, *Adaptive and Learning Agents*, 2012.

[34] H. Bou-Ammar, E. Eaton, P. Ruvolo, and M. E. Taylor. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In *AAAI*, 2015.

[35] 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.

[36] J. Kober. Learning motor skills: From algorithms to robot experiments. *it - Information Technology*, 56(3):141–146, 2014. doi:10.1515/itit-2014-1039.

[37] E. Rohmer, S. P. N. Singh, and M. Freese. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In *IROS*, 2013.

[38] Source code of ewerton et al. (2019). `https://github.com/marcoewerton`.

[39] S. James, M. Freese, and A. J. Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

[40] Source code of seker et al. (2019). `https://github.com/myunusseker/CNMP`.

[41] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2014.

[42] R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare. Safe and efficient off-policy reinforcement learning. In *NIPS*, 2016.

[43] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9:2579–2605, 2008.

A Implementation Details

For encoder and decoder model, neural networks with ReLU activations were used and they were trained using Adam optimizer [41]. In addition to given source codes, network details can be found in Table 3 for full reproducibility. Note that our focus was not on optimizing network sizes even though we observed that similar accuracies can be obtained using smaller networks. For reinforcement learning, we used Retrace algorithm [42] and in the transfer learning task, latent space visualization (Figure 7) was obtained using T-SNE method [43].

Table 3: Neural network details for different tasks. Encoder and decoder structure show number of neurons for each layer.

	Encoder Structure	Decoder Structure
Extrapolation in 2D Obstacle Avoidance (4.1)	128,64,32,16,8	124,124,2
Extrapolation in 2D Pushing Task (4.2)	128,128,64,32	128,128,128,6
Wall Avoidance (4.2)	128,128,128,64	128,128,64,32,4
Transfer Learning, Model of 3 Dof (4.3)	128,128,128,64	128,128,128,128,6
Transfer Learning, Model of 4 Dof (4.3)	128,128,128,64	128,128,128,128,8
Pick and Place (4.4)	128,128,64,32	128,128,128,12
Water Pouring (4.4)	128,128,64,32	128,128,128,2

B Further analysis of extrapolation amount

Recall that in section 4.1, we showed performance of ACNMP for extrapolation point in 2D obstacle avoidance task. In this analysis, we investigated to what extent our algorithm is successful by placing the extrapolation point further away from the demonstrations. Although the algorithm still attempts to find solutions that satisfy the constraints, the shapes of the generated trajectories change significantly (see Fig. 9). Note that given only one constraint, "what must the right shape of the complete trajectory be?" is not a well-defined question.

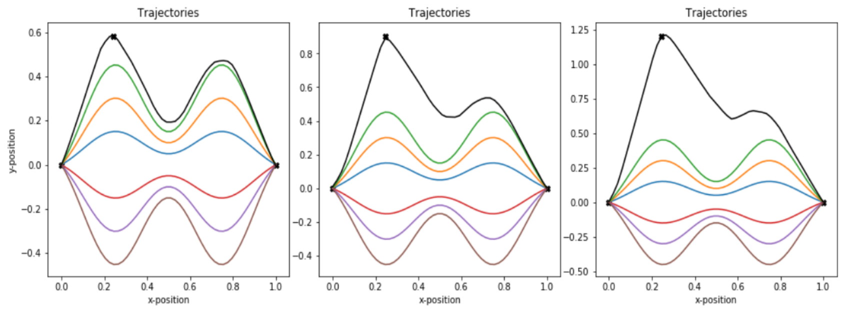

Figure 9: ACNMP performance in response to the increasing amount of extrapolation.Colored and black lines correspond to the demonstrated and generated trajectories, respectively.

C Demonstrations for wall avoidance experiment (4.3)

Temporal alignment: Recall that in section 4.3, we showed ACNMP's performance in a 2D simulation wall avoidance experiment where the position of the wall and goal point are changing. Importantly, in this experiment, we verify that the demonstrations do not need to be time-aligned. In figure 10, the demonstrations are illustrated. Note that the trajectory is traveled uniformly in time and avoiding the wall behavior should occur at different time steps. **Reward function**: [28] stated that the third component of reward was set as the minimum distance to the center of the hole but plots show different objective. After personal contact with the authors, the exact reward was clarified as stated in the text.

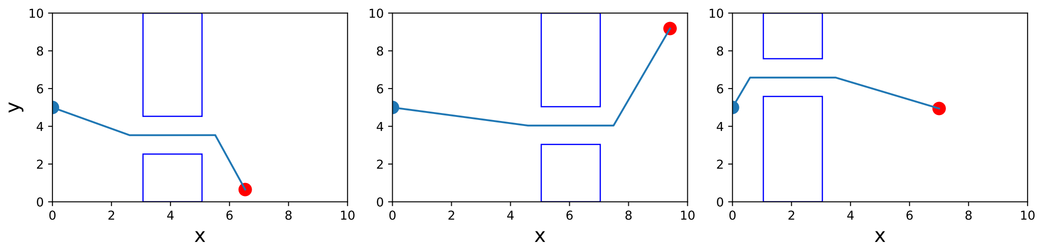

Figure 10: The demonstrations for the wall avoidance experiment are shown. The blue and red circles show the start and end points respectively.

D Joint trajectories for the first real robot experiment

Recall that in section 4.4, we showed performance of ACNMP for real world pick and place task, where six joints of UR10 robot were optimized to carry an object while avoiding an obstacle. Here, we would like to provide ACNMP joint trajectories for pick and place real robot task in figure 11. It can be seen that the found solution is similar to the expert solution in terms of shape, even though there is no reward for similarity.

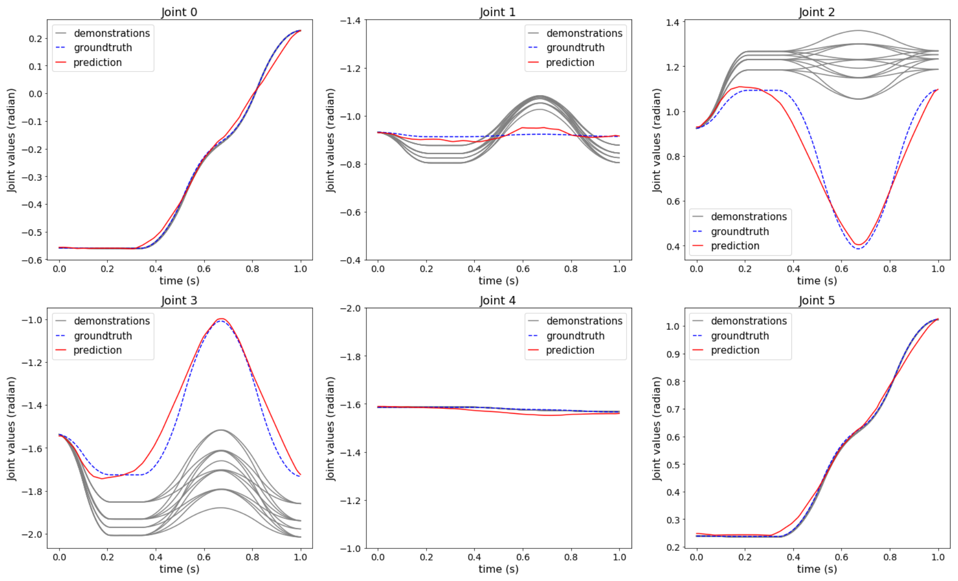

Figure 11: ACNMP predictions for pick and place task. Expert demonstrations are shown in gray, corresponding expert trajectories for extrapolation task parameters are shown in blue and ACNMP predictions are shown in red.