Summary

Bardia Mojra
January 26, 2022
Seminar on Continual Learning
Robotic Vision Lab

A Simple Approach to Continual Learning By
Transferring Skill Parameters

K.R. Zentner, Ryan Julian, Ujjwal Puri, Yulun Zhang, and Gaurav S. Sukhatme

Learning robotic manipulation tasks from vision is challenging and requires massive amount of data and hundreds of hours of training. In recent years, physics-based simulators and Deep Reinforcement Learning (DRL) algorithms have been widely used to allow learning agents to explore the solution space and construct rich function approximators or policy functions. In such end-to-end approaches, researchers train a Deep Neural Network (DNN) with both visual input and motor control output data provided at the same time but rather in short and similar episodes. Nonetheless, the main issue with this approach is the fact that it is still too computationally expensive to learn new skills from scratch or to even retrain after transfer learning.

The authors introduce a novel and more efficient method for continual learning of manipulation tasks by transferring skill parameters. They show that representing past experiences only in form of skill policies, methodical pretraining, and appropriately choosing when to transfer those skill policies is a simple yet effective recipe for building a continual learner in the context of robotic manipulation. New skill policies are learned based on *prior skills* or *skill libraries* to enable efficient acquisition and transfer of dynamic control policies. Similar to [1] and [2], they train reuseable skill libraries in simulation to develop composable skill policies in form of latent space parameter.

Setting They define continual learning problem as iterated transfer learning for multi-task reinforcement learning (MTRL) on a possibly-unbounded discrete space of tasks $\mathcal{T}$. S a single continuous state space shared among all tasks T. A a single continuous action space shared among all tasks T. The MTRL problem is defined by $(\mathcal{T},\ S,\ A)$ and each task $\tau \in \mathcal{T}$ is an infinite-horizon Markov decision process (MDP) defined as

$$\tau = (S, A, p_\tau(s, a, s'), r_\tau(s, a, s'))\,. \tag{1}$$

Tasks are differentiated only by their reward functions r_τ and state transition dynamics p_τ. Thus, for simplicity they define

$$\tau = (r_\tau, p_\tau)\,. \tag{2}$$

Importantly, the authors did not presume the robot has access to all tasks in T at once or even a representative sample. The robot can only access one task at a time. The time between task transitions is referred to as an 'epoch' with each having a unique index. Tasks may reapear and when solving a "target task," the robot can only skill policies acquired while solving prior tasks M (the "skill library"). Thus, they redefine the MTRL problem as

$$\tau = (S, A, M_i, p_{\tau_i}(s, a, s'), r_{\tau_i}(s, a, s'))\,, \tag{3}$$

where M_i represents the set is manipulation skills the robot is initialized with and i is the epoch number. They primarily use PPO [3] for the RL algorithm F that effectively samples the distribution of value function before training.

Simple Continual Learning with Skill Transfer

Algorithm 1 Proposed Continual Learning Framework

Input: Initial skill library $\mathcal{M}_0$, target task space $\mathcal{T}$, RL algorithm $\mathcal{F} \rightarrow (\pi, \rho)$, target task rule `ChooseTargetTask`, base skill rule `ChooseBaseSkill`
$i \leftarrow 1$
while not done **do**
  $\tau \leftarrow \texttt{ChooseTargetTask}(\mathcal{T}, \mathcal{M}_{i-1})$
  while π_{target} not solved **do**
    $\pi_{base} \leftarrow \texttt{ChooseBaseSkill}(\mathcal{T}, \mathcal{M}_{i-1})$
    $\pi_{target}, \cdot \leftarrow \mathcal{F}(\tau, \texttt{clone}(\pi_{base}))$
  end while
  $\mathcal{M}_i \leftarrow \{\pi_{target}\} \cup \mathcal{M}_{i-1}$
  $i \leftarrow i + 1$
end while
Output: Skill library $\mathcal{M}_i$

Figure 1: Algorithm 1

"Warm-Up" Procedure for Value Function Transfer In order to tune a skill on a task, they need a value function that estimates expected the reward of that skill policy on the task τ. They used a initial gradient batch sampling for estimating value function distribution similar to PPO. This allows them to tune the value function of any skill on the new task and was necessary to perform transfer effectively with PPO.

Rejecting Bad Since new skills are build upon prior skills in a hierarchial manner it is import to train and retain good transferred skills. The authors built-in a mechanism where the learning agent can ternimate a bad transfer at any time. It is characterized by a transferred policy π_{base} that "falls too far behind" the from-scratch policy.

Skill-Skill Transfer Cost In order to transfer skills more efficiently, they tansformed the continual learners training sequence form *Random Skill Tranfer* to *Skill Curriculum*. Furthermore, they developed a method for measuring skill-skill transfer cost by counting the number of samples needed to acquire a target Skill, starting from a given base skill.

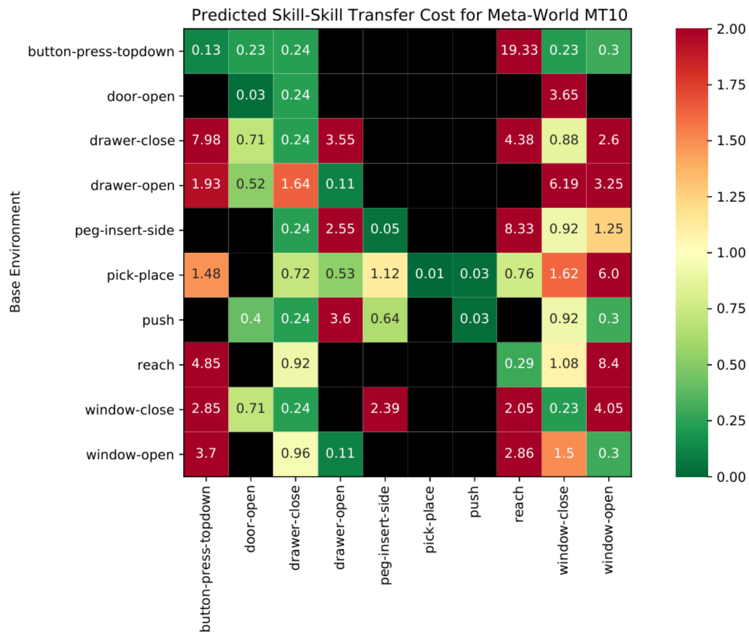

Figure 2: Predicted Skill-Skill Transfer Cost

Predicted Skill-Skill Transfer Cost They define as the ratio of time steps required to learn a task in MT10 [4] by skill transfer to learning the same task from scratch, using each other possible skill policy as a base skill. Equation (4) and figure (2) show skill transfer equation and scores, respectively.

$$A_{base \to target} = \frac{C_{base \to target}}{C_{scratch \to target}} \tag{4}$$

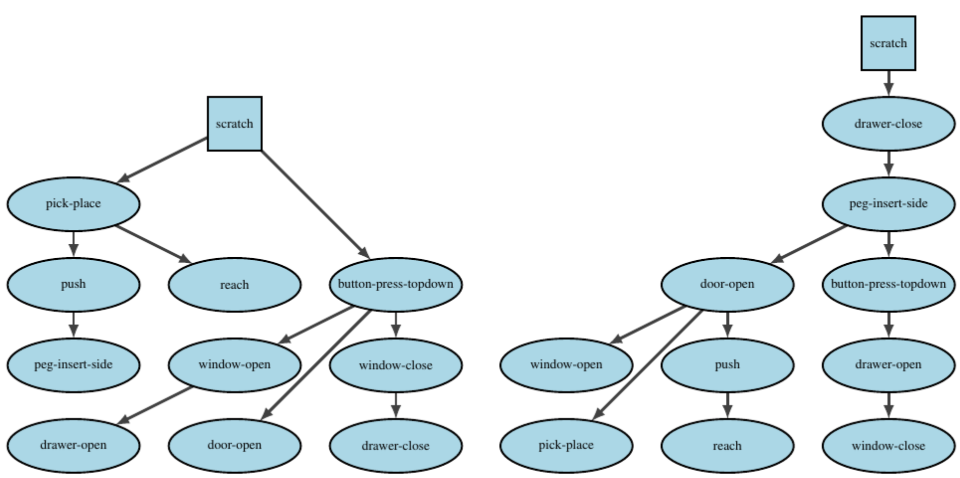

Figure 3: Optimal and Pessimal Trees

Algorithm 2 DMST-Based Curriculum Transfer

Input: Initial skill library $\mathcal{M}_0$, target task space $\mathcal{T}$, RL algorithm $\mathcal{F} \to (\pi, C)$
$V \leftarrow \mathcal{T} \cup scratch$
$E \leftarrow \{\}$
for $\tau_{base} \in \mathcal{T}$ **do**
  $E \leftarrow (scratch, \tau_{base}, -1.0)$
  $\pi_{base}, C_{scratch \to target} \leftarrow \mathcal{F}(\tau_{base}, \pi_{random})$
  for $\tau_{target} \in \mathcal{T}$ **do**
    $\cdot, C_{base \to target} = \mathcal{F}(\tau_{target}, \pi_{base})$
    $E \leftarrow E(\tau_{base}, \tau_{target}, C_{base \to target}) \cup E$
  end for
end for
$T_{optimal} \leftarrow \texttt{kruskal}((V, E))$
$i \leftarrow 1$
$\pi_{base} \leftarrow \pi_{random}$
for $\tau_{target} \in \texttt{traverse}(T_{optimal})$ **do**
  while τ_{target} not solved **do**
    $\pi_{target}, \cdot \leftarrow \mathcal{F}(\tau_{target}, \texttt{clone}(\pi_{base}))$
    if τ_{target} not solved **then**
      $E \leftarrow E \setminus (\tau_{base}, \tau_{target})$
      $T_{optimal} \leftarrow \texttt{kruskal}((V, E))$
    end if
  end while
  $\mathcal{M}_i \leftarrow \{\pi_{target}\} \cup \mathcal{M}_{i-1}$
  $i \leftarrow i + 1$
end for
Output: Skill library $\mathcal{M}$

Figure 4: DMST-Based Curriculum Transfer Learning Algorithm

Curriculum Selection Algorithm They generate a skill-skill transfer cost for each task $\tau \in \mathcal{T}$, which form the weighted adjacency matrix of a densely-connected directed graph with skill-skill transfer costs as the directed edge weights. With this problem setting, they solve for the Directed Minimum Spanning Tree (DMST) to obtain the optimal path for sequentially transferring all tasks. Thus, they create an optimal transfer learning curriculum for transferring and retuning any skill. Figure (3) and (4) depict examples of optimal and pessimal trees and DMST-based curriculum transfer learning algorithm, respectively. Figure (5) shows performance comparison between different skill curriculums for MT10.

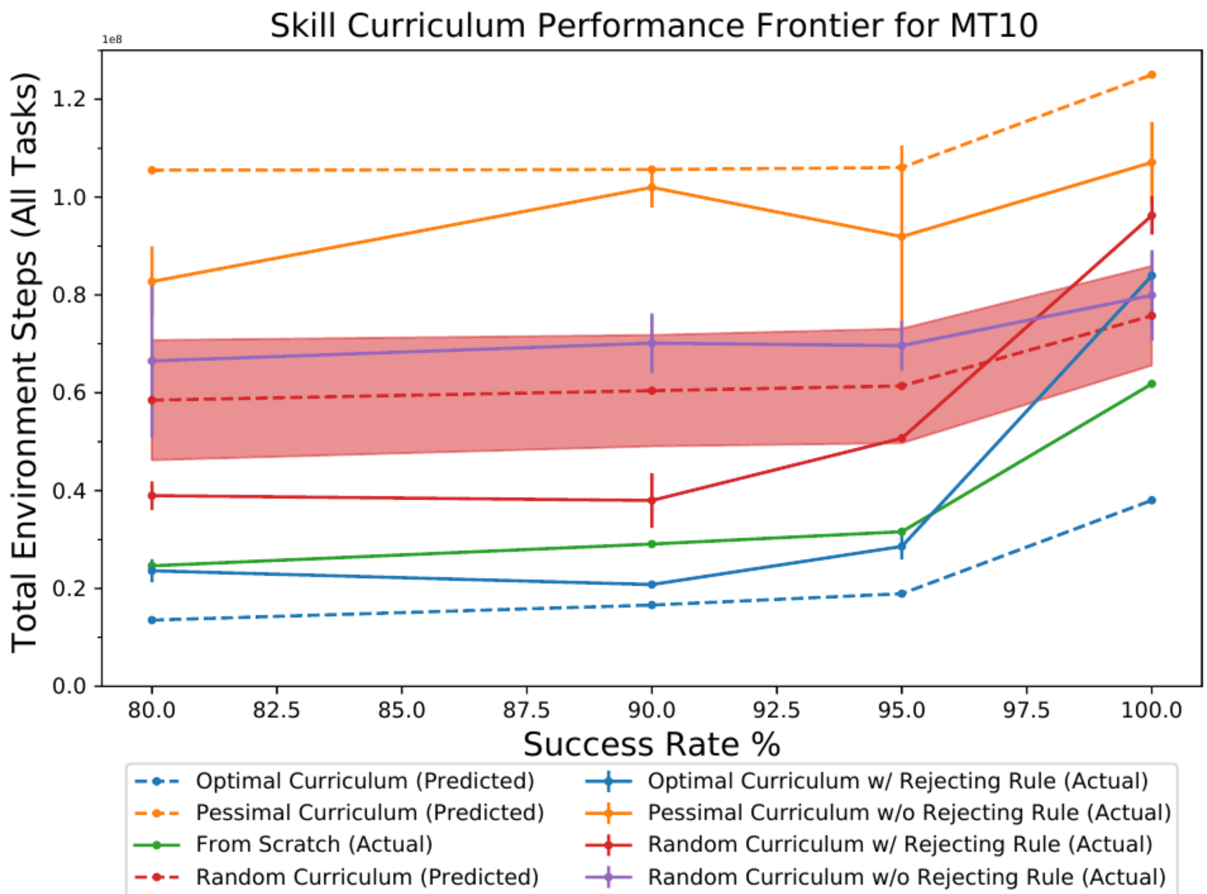

Figure 5: DMST-Based Curriculum Transfer Learning Algorithm

References

[1] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *International Conference on Learning Representations*, 2018.

[2] R. Julian, E. Heiden, Z. He, H. Zhang, S. Schaal, J. Lim, G. Sukhatme, and K. Hausman, "Scaling simulation-to-real transfer by learning composable robot skills," in *International Symposium on Experimental Robotics*, pp. 267–279, Springer, 2018.

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

[4] T. Yu, D. Quillen, Z. He, R. C. Julian, K. Hausman, and C. Finn, "577 sergey levine. metaworld: A benchmark and evaluation for multi-task and meta reinforcement 578 learning," 2019.