Learning Action Priors for Visuomotor transfer

Anurag Ajay [1] **Pulkit Agrawal** [1]

Abstract

Reinforcement learning (RL) systems have proven to be successful in the domain of games, but have met with limited success on real-world tasks. A significant reason is that RL systems are extremely sample-inefficient. They are usually designed to solve one particular task from scratch instead of reusing the knowledge gained by solving previous tasks. Prior works have tried to resolve this issue by learning a prior over past state distribution in form of a pre-trained policy. However, these approaches fail when new tasks have different state distribution and different reward function. In this work, we propose to learn an additional prior over useful action trajectories, called extended action prior (EAP). We show that EAP leads to significant performance boosts in the context of transfer learning across tasks and in multi-task reinforcement learning. Importantly, the proposed method is simple and agnostic to the specific choice of learning architectures and RL algorithms. We evaluate our method on meta-world and procgen benchmarks.

1. Introduction

Reinforcement Learning (RL) systems have achieved impressive performance in games such as GO, ATARI, and many others (Mnih et al., 2015; Silver et al., 2017). Yet, their applicability to real-world tasks remains limited due to the appalling data requirements. A significant reason for this gap in data efficiency is that current RL agents are designed to solve one particular task from scratch. In contrast, humans make use of their experience to quickly learn new tasks. The typical method for transferring information in deep RL is to finetune the policy network pre-trained on a previous set tasks on a new task at hand. This process essentially transfers information about what actions to perform if the observations in the new task are similar to those in the previous tasks. However, in many real-world scenarios a new task might have a different reward structure or there might be significant differences in say the visual observations. In such scenarios, simply fine-tuning the policy is unlikely to work well.

Our insight is that only transferring observation specific information about how to act is limited. To see why, consider for instance video games, by jumping up and down at one spot, the agent will not succeed. However, jumping from one platform to another or over an obstacle increases the chance of success. This example suggests that not all sequences of actions are important, but there is possibly a relatively small set of action sequences (or skills) that are critical for success. In other words, in addition to observation-specific information that can be transferred, there is also structure in the action space that is shared between tasks. Current methods for transfer in RL do not exploit this structure in the action space. As an illustration, consider using a pre-trained policy to solve a new task with a different reward structure, which otherwise has similar sensory statistics as the previous tasks. To compensate for the changes in the reward structure, the agent might need to unlearn its current strategy and learn a new sequence of actions. This search will be more efficient if the search space is constrained only to the set of useful skills. Similarly, if the statistics of the observation space change drastically, the agent will have to learn from random exploration. Again, exploring in the space of useful skills will be beneficial. We therefore propose a simple, yet highly effective method for transferring *sensorimotor* knowledge across tasks instead of just transferring the shared structured in the sensory space.

The natural next question is how to exploit the structure in the action space? In a spirit similar to prior work on skill learning (Bacon et al., 2016; Lynch et al., 2019), we propose to condense successful trajectories on previous tasks into a set of continuously parameterized skills. Specifically, we learn a latent space (z_t) for sampling action sequence, i.e. $a_{t:t+T} \sim z_t$. The latent space, z_t, which we call as the *Extended Action Prior* (EAP), is optimized to sample action sequences from successful trajectories on previous tasks with high-probability. Suppose that the pretrained policy is represented by $\pi(a_t|s_t;\theta)$, where s_t is the current state and θ denotes the policy parameters. Using EAP, this policy can

[1]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, USA. Correspondence to: Anurag Ajay <aajay@mit.edu>.

International Conference on Machine Learning workshop on Inductive Biases, Invariances and Generalization in RL (BIG), Copyright 2020 by the author(s).

be transformed into $\pi(z_t|s_t;\theta)$. Finetuning on a new task with EAP, therefore, ensures that we transfer *sensorimotor* and not just sensory information. We show that finetuning with EAP leads to substantial improvements in performance when transferring to a task with a different reward structure and visual appearance, on video game environments released as part of the procgen benchmark (Cobbe et al., 2019). Note that in contrast to prior work that has used skills/options to improve learning efficacy on a single task, here we are advocating for using skills to transfer action priors across tasks. A detailed discussion comparing ours and prior work is provided in Appendix A.

Until now we have assumed that the agent had access to successful trajectories from previously solved tasks. While it is possible to train on different tasks individually, it turns out that learning simultaneously on multiple tasks remains an open problem in RL (Cobbe et al., 2019; Schaul et al., 2019; Yu et al., 2020). This is undesirable because training separate policies for different tasks means that it is not possible to share common information between tasks. The difficulty of learning from multiple tasks in RL arises due to exacerbation of variance in gradients. The typical source of variance is that a certain set of actions might be present in both trajectories that achieve high and low rewards. In turn, this leads to conflicting gradients which leads to instabilities in training. In case of multiple tasks, a certain action sequence may lead to high-reward for one task, but lower reward for another task. Such scenarios further increase the variance in gradients leading to learning failures in multi-task RL setups.

A recent work proposed to mitigate this noisy gradient problems by explicitly removing conflicting information between gradients computed from different tasks (i.e., projecting conflicting gradients or pc-grad; (Yu et al., 2020)). Complementary to this approach, we make an observation that problem of noisy gradients can also be mitigated by improving exploration. This is because, as explained above, the noise/variance problem is caused by the same set of actions appearing in different trajectories that result in high and low rewards respectively. If the agent is performing random exploration, just by chance, it is more likely for an action sequence to be part of both high and low reward trajectories. In contrast, if the exploration is structured, chances of such occurrence will reduce. We empirically demonstrate that this intuition is indeed true and better exploration leads to large gains in performance in multi-task RL. Specifically, we show that multi-task learning with EAP on on MT10 and MT50 benchmarks of the metaworld (Yu et al., 2019) environment substantially outperforms learning without EAP.

2. Method

Our aim is to learn action and state priors from past experiences for improved transfer and performance. Let's say

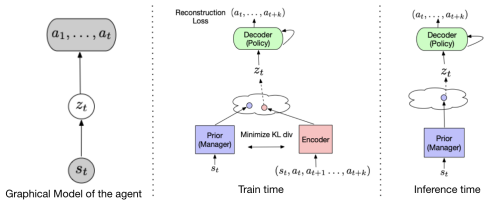

Figure 1. Extended Action Prior (EAP). **Left** shows the underlying graphical model of the agent where states and action sub-trajectories are observed. **Center** shows execution of EAP during train time while **right** shows execution of EAP during inference time.

we already have a dataset of high-reward trajectories collected by solving some prior tasks. We use the dataset to learn priors which gives high likelihood to the trajectories in the dataset. Let $D = \{(s_t^i, a_t^i)_{t=1}^T\}_{i=1}^N$ where T is the task horizon and N is the number of trajectories. We want to learn an action prior π_{act} which takes in an extended action z_t and generates the action sub-trajectories $(a_{t'})_{t'=t}^{t+k}$ where k is a hyperparameter. In addition, we want to learn a state prior π_{state} which takes in the current state s_t and generates an extended action z_t that has high likelihood of generating a sub-trajectory from the dataset when used with the action prior. Overall, we want to learn π_{act} and π_{state} such that $E_{s_t,(a_{t'})_{t'=t}^{t+k}}[E_{z_t\sim\pi_{state}(s_t)}[\pi_{act}((a_{t'})_{t'=t}^{t+k}|z_t)]]$ is maximized. We use π_{state} and π_{act} together to explore new tasks. When we encounter a new task with different state distribution or reward function, π_{state} might not give the right extended actions but π_{act} will still generate a sub-trajectory that has high-likelihood of coming from the dataset (i.e. has been useful in the past).

2.1. Extended Action Prior (EAP)

In this subsection, we will describe the details of our proposed model Extended Action Prior (EAP). EAP is based on framework of conditional VAE (Sohn et al., 2015). We assume that the agent observes observes the current state s_t, generates a latent extended action z_t based on the current state and then unrolls the extended action z_t in form of a sub-trajectory $(a_t, a_{t+1}, \ldots, a_{t+k-1})$. This assumed graphical model is visualized in Figure 1. Under this graphical model, the encoder uses the current state and the action sub-trajectory to infer the latent extended action and the decoder (i.e. π_{act}) uses the extended action to reconstruct the sub-trajectory. In addition, we learn a prior (i.e. π_{state}) which generates the extended action only using the current state. We will now describe these components in detail.

Prior is our state prior π_{state} and acts like a manager policy which takes in the current state and gives the distribution of extended action from which we sample the extended action during the inference time. $(\mu_t^{pr}, \sigma_t^{pr}) = \pi_{state}(s_t)$; $z_t \sim \mathcal{N}(\mu_t^{pr}, \sigma_t^{pr})$. **Decoder** is our action prior π_{act} and acts

like a low-level policy which takes in the extended action and unrolls a corresponding sequence of k action distribution from which we sample the action sub-trajectory. $(\pi_t, \ldots, \pi_{t+k-1}) = \pi_{act}(z_t)$; $(a_t, \ldots, a_{t+k-1}) \sim (\pi_t, \ldots, \pi_{t+k-1})$. **Encoder** takes in the current state and the action sub-trajectory and infers the distribution of extended action which could have generated the action sub-trajectory. We sample the extended action from this distribution during the train time. $(\mu_t^{enc}, \sigma_t^{enc}) = Encoder(s_t, a_t, \ldots, a_{t+k-1})$; $z_t \sim \mathcal{N}(\mu_t^{enc}, \sigma_t^{enc})$. EAP is trained from expert trajectory data $D = \{(s_t^i, a_t^i)_{t=1}^T\}_{i=1}^N$ by optimizing the following loss function $L = L_{KL} + \lambda L_\pi$ where $L_{KL} = KL(\mathcal{N}(\mu_t^{enc}, \sigma_t^{enc}) || \mathcal{N}(\mu_t^{pr}, \sigma_t^{pr}))$ and $L_\pi = -\frac{1}{k}\sum_{i=t}^{t+k-1} \pi_i(a_i|z_t)$. L_{KL} forces the extended action distribution generated by the prior to place high probability on extended actions inferred by the encoder from the expert trajectory data. L_π forces the decoder to maximize the action likelihood of the expert sub-trajectories. Note than k and λ are hyper-parameters. During the inference time, the prior takes in the current state s_t and produces an extended action z_t which gets unrolled into an action sub-trajectory $(a_t, \ldots, a_{t+k-1})$ by the decoder. Thus, EAP takes in the environment feedback every k time steps. We visualize the EAP model in figure 1.

2.2. Transfer to new tasks

The decoder of the EAP π_{act} captures the trajectory distribution that was useful for solving the task on which it was trained. Furthermore, the decoder is dependent on the state only through the extended action. Therefore, when given a new task that requires the same strategy, we can just finetune the prior or learn a new policy on top of the decoder to quickly learn the new task. Intuitively, the learning problem becomes easier because exploration is more structured as we are exploring in space of useful trajectories and the task horizon gets reduced by a factor of k. As shown in section 3, learning policies with EAP leads to faster transfer in both visual and non-visual domains even when the new task has different state distribution and different reward function. The only assumptions we make are that the tasks should have the same action space and require similar strategy so that they have the same distribution of useful trajectories.

3. Experiments

We will empirically validate the following claims: (1) Since EAP captures useful structure in the action space, learning with EAP results in faster and better transfer to new tasks. (2) Learning with EAP improves exploration which leads to substantial improvements in multi-task learning.

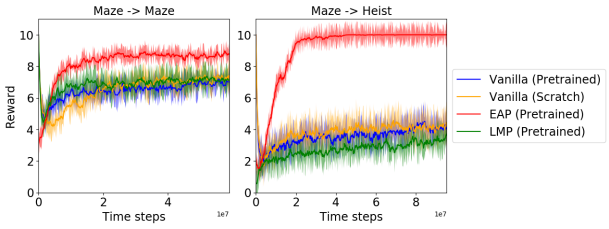

Figure 2. With improved exploration, policy learned with EAP learns faster on visually new domains, new levels (100-200 levels) of maze and heist (0-100 levels), and achieves a better performance when compared to vanilla policy (either pretrained or learned from scratch) and LMP policy. We pre-train vanilla policy, LMP policy and EAP on 0-100 levels of maze. The dark lines represent mean reward over 3 independent seeds. The shaded area represent one standard deviation interval.

3.1. Learning policies with EAP leads to faster and better transfer

We evaluate transfer performance in three settings: (a) the dynamics and visual statistics are preserved between tasks, but the reward function changes; (b) the dynamics and reward structure remains the same, but visual statistics change; and finally (c) scenarios where dynamics, reward function and visual statistics all change. This evaluation is conducted using six games (*maze, heist, ninja, coinrun, jumper, climber*) from the procgen (Cobbe et al., 2019) environment suite. For each game, procgen provides thousands of levels. Different levels of the same game have the same dynamics and reward structure, but are visually different. The three settings described above can be created by either transferring between different levels of the same game or by transferring across games. In each scenario, EAP and the baseline methods are pre-trained on hundred levels of the same game. The pre-trained policies are then finetuned using PPO (Schulman et al., 2017) on 100 test levels.

We compare the performance of our method, EAP, against: (1) *Vanilla (pretrained)*: If EAP indeed captures useful action priors, then it should outperform a *flat* policy that explores in the space of all actions. Transferring with *vanilla (pretrained)* amounts to finetuning $\pi(a_t|s_t)$, instead of the EAP policy which is finetuning $\pi(z_t|s_t)$. (2) *Vanilla (scratch)*: If a policy trained from scratch on testing levels outperforms a pre-trained policy, it indicates that pre-trained policy did not capture any transferable information. (3) *LMP* (Lynch et al., 2019): We also compare against a previously published method for learning skills, also described in Appendix B.

Transfer to tasks with different reward structure: The *maze* game (see Figure 2) requires the agent to navigate to a goal location in a 2D grid world. A sparse reward of +1 is provided when the agent reaches the goal. The *heist* game has similar visual statistics and dynamics as *maze*, but has a

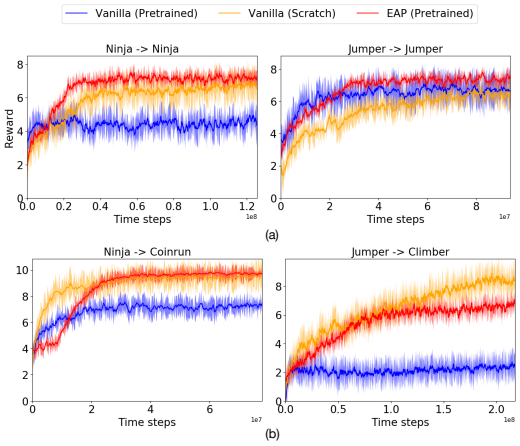

Figure 3. Policies learned with EAP consistently learns faster than (pretrained) vanilla policies on visually new domains: (a) new levels (100-200 levels) of the same game (b) new game (0-100 levels). We pre-train vanilla policy and EAP on 0-100 levels of a game. The dark lines represent mean reward over 3 independent seeds. The shaded area represent one standard deviation interval.

different reward structure. Here the agent needs to collect keys to open locks before it can get to the goal. The reward is again sparse, making it significantly more challenging than the *maze* game.

Results in Figure 2 show two things: (a) EAP is noticeably better than baseline methods when transferring to new levels of the *maze* environment (left column) and (b) EAP substantially outperforms baselines when transferred to *heist*. These results convincingly demonstrate that opposed to simply transferring visual representations (i.e., *Vanilla (pretrained)*, transferring visuo-motor information using action priors helps the agent easily adapt to new reward structures. Quite interestingly, performance of *LMP* is comparable to *Vanilla (pretrained)*. This suggests that *LMP* fails to capture any useful action priors. As discussed in Appendix B, *LMP* is prone to posterior collapse when learning in environments such as *maze* that have a single optimal strategy. In contrast, EAP doesn't suffer from this problem and we expect *LMP* and EAP to perform similarly in scenarios where there are multiple optimal strategies (i.e., multi-modality) . Due to inferior performance of *LMP*, we do not include comparisons to it in the subsequent experiments.

Transfer to tasks with different visual appearances: Different levels of the same game in *procgen* are designed to have visual differences, but have the same dynamics and rewards. One of our hypotheses was that in such scenarios, exploring with action priors will lead to better transfer in comparison to transferring only visual priors. Results of transfer between different levels of *maze* shown in Figure 2 (left column) already confirmed our hypotheses. To ensure generality of the results, we also evaluated various methods on the games of *ninja* and *jumper*. Results in Figure 3(a)

Models	Average Success rate	
	MT10	MT50
PPO	0.15 ± 0.05	0.05 ± 0.02
PPO + EAP (ours)	0.7 ± 0.04	0.45 ± 0.03
SAC*	0.39	0.28
BC	0.84 ± 0.01	0.67 ± 0.01
Average of experts	1.0	0.82

Table 1. Due to improved exploration, EAP outperforms vanilla policy on MT10 ands MT50. We also provide two oracles: Average of experts and Behavioral cloning (BC). Methods with * are taken from other papers. The standard deviations is calculated over 3 independent seeds.

show that EAP indeed outperforms the baselines. It is interesting to observe that training from scratch outperforms the vanilla pre-trained policy at transferring across levels of *ninja*. This shows that there are significant visual differences between different levels of *ninja*. Despite these differences, EAP leads to good transfer.

Transfer to tasks with different rewards, dynamics and visuals: Finally, in Figure 2(b) we show transfer performance across games. The policies pre-trained on *ninja* and *jumper* are transferred to *coinrun* and *climber* respectively. While all four of these are platform games, there are significant differences. For example, in *coinrun* the agent needs to advance to the rightmost point to get to the goal, whereas in *climber* agent mostly needs to climb up and in *ninja* the agent must also shoot the enemies. Due to lack of significant shared structure between games, transferring with EAP does not results in performance improvements when compared to training from scratch. However, EAP does outperform vanilla policy by a big margin. While we are able to transfer information about useful actions, to achieve good transfer across different games the agent must realize that jumping on platforms, over enemies etc. are useful sub-tasks. However, current RL agents lack this ability and this is an exciting area for future research.

3.2. Training policies with EAP improves performance on multi-task RL

To investigate advantages of EAP in context of multi-task learning, we pre-train EAP using successful trajectories from goal-conditioned pick and place task. We chose this task as pick-place is a basic operation for performing more complex manipulation tasks that are part of metaworld. Results in Table 1 show that PPO + EAP outperforms both PPO and Soft Actor Critic (SAC) (Haarnoja et al., 2018) on MT10 and MT50. While EAP outperforms other RL methods, it achieves about 70% of the accuracy obtained by training a policy with behavior cloning (BC). Training with BC essentially overcomes the exploration problem and is thereby essentially an upper bound (i.e., oracle) for the EAP performance. We detail ablation studies in the Appendix C.