Efficient Task Adaptation by Mixing Discovered Skills

Anonymous Authors[1]

Abstract

Unsupervised skill discovery is one of the approaches by which the agent learns potentially useful and distinct behaviors without any explicit reward. The agent is then expected to quickly solve downstream tasks by properly using a set of discovered skills rather than learning everything from scratch. However, it is non-trivial to optimally utilize the discovered skills for each task, which can be viewed as a fine-tuning method, and this has been less considered in the literature in spite of its importance. In this paper, we compare some fine-tuning methods showing how they inefficiently utilize the discovered skills and also propose new methods, which are sample-efficient and effective by interpreting the skills as a perspective of how an agent transforms the input state.

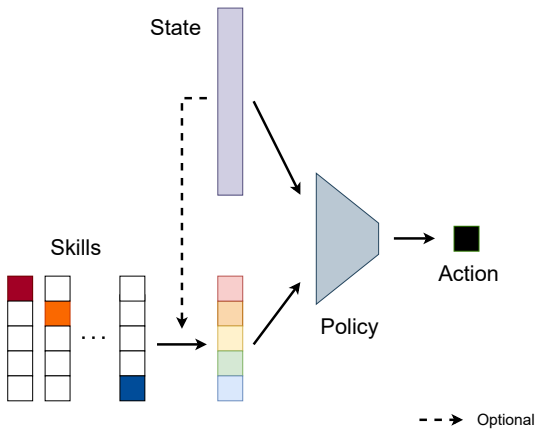

Figure 1. Overview of methods to mix a set of learned skills in fine-tuning phase. It is optional whether the current state affects how the skills combine together.

1. Introduction

Unsupervised learning for deep reinforcement learning as a pretraining phase is challenging in the sense that an agent should interact with the world and collect data to obtain useful information without any explicit reward. That way, the agent is supposed to gather information from the environment and learn representation of visited states, dynamics, and behaviors. When a downstream task is given, the agent is expected to quickly adapt to it by leveraging the learned representation, and this is considered as a fine-tuning phase. Recent works have shown that skill discovery is a promising way as a pretraining phase to learn diverse behaviors or skills, which potentially can help the agent quickly achieve a goal (Salge et al., 2014; Gregor et al., 2016; Eysenbach et al., 2018), in an unsupervised manner.

Although the agent could learn diverse skills, it is not obvious at a glance how the agent manages them to solve a downstream task. Since each skill is represented by a separate policy, the agent has to accordingly handle multiple policies to properly take actions in a typical reinforcement learning framework. DIAYN (Eysenbach et al., 2018) suggests a fine-tuning method that measures the performance of all discovered skills to find the best one for the given downstream task, which might be expensive as the total number of skills grows, and fine-tune it. Meanwhile, the recent work (Laskin et al., 2021) suggests a simpler method that randomly samples a single skill to be used for some interval and uses multiple skills during fine-tuning. However, both fine-tuning methods show poor performance compared to the agent that learns everything from scratch. Instead of exhaustive skill search or naive skill sampling, we propose new fine-tuning methods to efficiently and effectively utilize the discovered skills.

To properly understand how to combine discovered skills, we first take a look at an instructive example based on DIAYN. Each skill is represented by a one-hot vector z, and it is used to derive the corresponding policy $\pi(a|s, z)$. The state representation $\tilde{s} = Ws$ is simply transformed (shifted) into different skill-spaces according to the selected skill

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the First Workshop of Pretraining: Perspectives, Pitfalls, and Paths Forward at ICML 2022. Do not distribute.

representation $\tilde{z} = Uz$. That way, we consider the skill representation $\tilde{z}$ as an agent's *perspective* and examine whether the agent can benefit from combining different skill-wise perspectives and manipulating the state representation in some shared skill-spaces.

Therefore, we mainly focus on how to combine a set of discovered skills. By formulating the policy in a specific way, we derive skill fusion (mixing) methods in a state-agnostic manner, which combines skills independent to the input state, and a state-aware manner that uses the information of the input state.

2. Preliminaries and Related Work

Background. We consider a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p)$ without an external reward function for a typical unsupervised reinforcement learning. $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, and $p(s'|s,a)$ is a dynamics. Given a skill z sampling from a skill distribution $p(z)$, a policy $\pi(a|s,z)$ is learned in option framework.

Skill discovery. In unsupervised RL setting, *skill discovery* means to learn diverse behaviors with an intrinsic reward such as *empowerment* (Salge et al., 2014). This is formulated by maximizing mutual information between states and skills $\mathcal{I}(S;Z) = \mathcal{H}(S) - \mathcal{H}(S;Z) = \mathcal{H}(Z) - \mathcal{H}(Z;S)$, where $\mathcal{H}(\cdot)$ is the entropy in information theory. There are a number of previous methods to optimize the objective above. VIC (Gregor et al., 2016) and DIAYN (Eysenbach et al., 2018) consider the final state and every state, respectively, in order to discriminate each skill. The better the discriminator distinguishes each skill, The more intrinsic rewards the agent obtains. Thanks to the discriminator, the agent is motivated to learn distinct skills to maximize intrinsic reward. HIDIO (Zhang et al., 2021) learns high-level policy choosing skills in a hierarchical manner. APS (Liu & Abbeel, 2021) leverages successor features to learn diverse behaviors. DADS (Sharma et al., 2019) focuses on the predictability of behaviors and learning dynamics.

After the skill discovery (pretraining) phase, an agent should adapt to the task with learned skills. A simple but general method is to select one skill with the initially highest reward for each task and further fine-tune this skill (Eysenbach et al., 2018; Laskin et al., 2022). Some methods train an additional high-level policy to choose skills to utilize more than just one skill for each task in a sequential way (Sharma et al., 2019). For a specific purposed task such as goal-reaching, there are zero-shot skill adaptation methods (Sharma et al., 2019; Choi et al., 2021). We propose the general but effective fine-tuning methods that achieve sample efficiency and high performance.

3. Proposed Method

The main idea of our approach is to combine the learned skills to utilize for downstream tasks. We acquire skills in the pretrain phase following DIAYN (Eysenbach et al., 2018) framework which is generally used as a skill discovery baseline.

3.1. Understanding Skill Fusion

We note that the skill z is a k-dimensional discrete random variable sampled from the distribution $p(z)$. Then we can write a policy as

$$\pi(a|s) = \mathbb{E}_{z\sim p(z|s)}[\pi(a|s,z)]. \tag{1}$$

One can fine-tune a *controller* $p(z|s)$ as in a hierarchical RL framework (Sharma et al., 2019), or simply sample skill from $p(z)$ (Laskin et al., 2021) under the assumption that skill distribution is independent to the state distribution. In both cases, skill z is first sampled from $p(z)$ or $p(z|s)$, and action a is derived from the policy depending on the skill. The limitation of those methods is that they just sequentially perform skills rather than combine and mix the skills in an underlying representation level.

Instead, we rather formulate the policy as

$$\tilde{\pi}(a|s) = \pi(a|s, \mathbb{E}_{z\sim p(z|s)}[\psi(z)]) \tag{2}$$

for some representation function ψ. In this work, we fix ψ as a one-hot vector representation of each discrete skill z in a natural manner. In the following sections, we introduce state-agnostic perspective fusion and state-aware perspective fusion depending on whether skill z is sampling from $p(z)$ or $p(z|s)$.

3.2. State-agnostic Perspective Fusion

We first consider the method to sample skills from $p(z)$ regardless of the input state. The simplest way of fine-tuning a policy is to choose one best skill, $p(z)$ is fixed as Dirac delta, as in the original DIAYN paper. However, it utilizes just one skill for a task which has a clear limitation when the task is complicated that only one skill is not enough, also, it is often expensive to know which skill is best-performing especially when the skill set is large.

Now we propose two state-agnostic fusion methods to alleviate these issues. First, we fix $p(z)$ to be uniform distribution, which combines the learned skills in the same weight. This simply takes advantage of mixing every skill without any additional parameters. Second, we train $p(z)$ as a categorical distribution, with additional trainable parameter w_i for $\mathbb{E}_{z\sim p(z)}[\psi(z)] = \sum_i w_i \psi(z_i)$, where $\sum_i w_i = 1$ and $w_i \geq 0$. The latter is generalized version of the original DIAYN and the former method.

Table 1. Fine-tuning method for skill discovery algorithm. Class shows whether the skill of each method is sequentially performed or combined. Each method samples the skill z from either $p(z)$ or $p(z|s)$, and finetune either only policy or all. We compare our methods to two previous works, *URLB DIAYN* (Laskin et al., 2021) and *Original DIAYN* (Eysenbach et al., 2018).

CLASS	METHOD	$p(z)$	$p(z\|s)$	FINETUNE
SEQUENTIAL	URLB DIAYN	UNIFORM	-	$\pi(a\|s,z)$
SEQUENTIAL OR FUSION	ORIGINAL DIAYN	DIRAC DELTA	-	$p(z)$ & $\pi(a\|s,z)$
FUSION	SAME IMPORTANCE	UNIFORM	-	$\pi(a\|s,z)$
FUSION	SIMPLE IMPORTANCE	CATEGORICAL	-	$p(z)$ & $\pi(a\|s,z)$
FUSION	SCRATCH CONTROLLER	-	CATEGORICAL	$p(z\|s)$ & $\pi(a\|s,z)$
FUSION	DIAYN CONTROLLER	-	CATEGORICAL (INIT.)	$p(z\|s)$ & $\pi(a\|s,z)$

We can interpret the representation level skill mix as follows. Since the skill z is represented as a one-hot vector, the weight parameters in a policy are switched on up to the location of 1 in a skill vector. Therefore, if all positions of a skill vector are non-zero, all weights are activated and more diverse representations are obtained. We can view this as the same state from different perspectives through skill.

3.3. State-aware Perspective Fusion

In the method above, skill is combined without considering the input state. However, we expect that better performance can be achieved if the combination of skills is changed adaptively according to the state. While it is possible to train the controller $p(z|s)$ from scratch, we propose a method to reuse a skill discriminator $q_\phi(z|s)$ trained in the pretraining phase. In DIAYN, the skill discriminator $q_\phi(z|s)$ predicts which skill is in charge of the input state so that each skill visits distinct states. Therefore, we can guide the agent in the natural direction rather than randomly combining skills by using a discriminator as an initializer of the controller. We show in the next section that although the state-aware fusion method is not as sample efficient as the state-agnostic method, it finally achieves a better performance.

4. Experiments

We evaluate our methods on URLB (Laskin et al., 2021), which contains three continuous control environments and twelve downstream tasks. The environments include 6-DOF *Walker*, 12-DOF *Quadruped*, and 9-DOF *Jaco arm*.

We follow the same evaluation process to URLB. We pretrain each agent without explicit rewards, then finetune for the downstream task. In the case of skill discovery method, the agent learns a set of useful behaviors which we call skills. Then the agent uses these skills to achieve higher cumulative rewards quickly in fine-tuning. When we evaluate our methods, we fix the pretrained agent to DIAYN based on DDPG (Lillicrap et al., 2016) and finetune it in various ways.

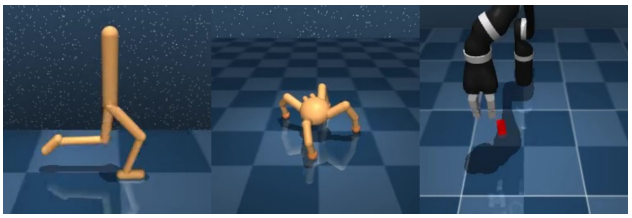

Figure 2. Continuous control environments from DeepMind Control Suite. Left: Walker, center: Quadruped, right: Jaco arm.

4.1. Sample-efficiency and Final Performance

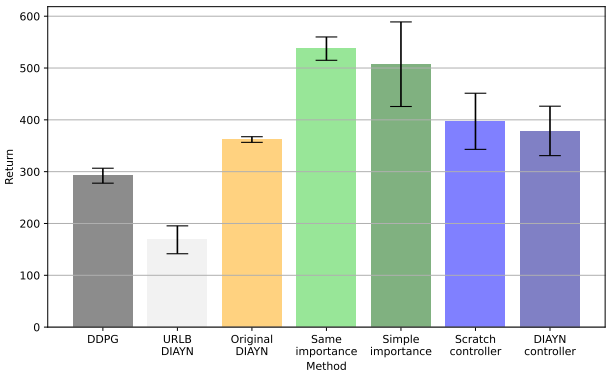

Figure 3. Results of 100k steps finetuning of skill adapting methods on the task Walker Run. Simple importance and Same importance method shows better sample efficiency than the others.

We compare our methods to three baselines, *DDPG* trained from scratch, DIAYN implemented in URLB (*URLB DIAYN*) and DIAYN proposed in the original paper (*Original DIAYN*). For *Original DIAYN*, we did not count the steps for selecting the best skill in order to make a baseline more challenging. Also, we choose the task Walker-run for the comparison because it is known as the most challenging task among the twelve downstream tasks in URLB (Laskin et al., 2021). We fine-tune each method for 100k steps. The

experiment in other environments is shown in section 4.2.

Same importance and *Simple importance* are both state-agnostic fusion methods, but the former fixes $p(z)$ and latter finetunes. *DIAYN controller* initializes a controller $p(z|s)$ with DIAYN skill discriminator, but *scratch controller* does not. The detail of our methods is in Table 1.

We observe the following results in Figure 3. *URLB DIAYN* underperforms other methods even including DDPG trained from scratch, showing that it does not fully use the potential of learned skills, and suggesting that fine-tuning is critical to measure the performance of the skill discovery method. On the other hand, both of our state-agnostic methods perform well, showing that the simple skill mix methods help to achieve high return quickly. The performance of state-aware methods is worse than state-agnostic methods and similar to the baseline (*Original DIAYN*) at 100k step. We presume training additional parameters for controller disturbs the fast adaptation.

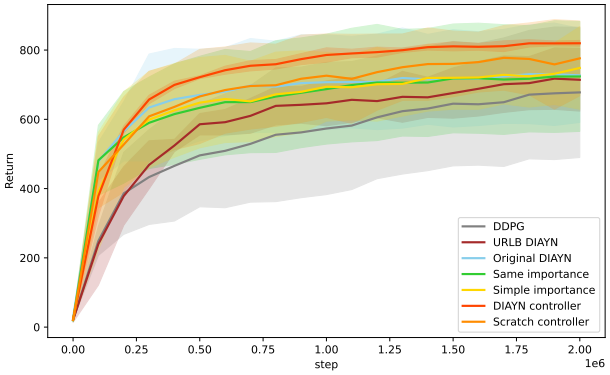

Figure 4. 2M steps finetuning results on *Walker run* task. State-agnostic methods shows fast learning, but state-aware method achieves higher return at the end.

We also compare our methods for the final performance. We show in figure 4 that state-aware fusion method outperforms state-agnostic fusion at some point. Followed by a certain amount of steps, the capacity of state-aware methods works. Moreover, we observe that *DIAYN controller* outperforms the *Scratch controller*. This shows that the DIAYN discriminator trained at pretrained stage is a good initializer for the skill controller. This transferred weight not only helped better performance at the final step, but also helped robust training which has lower variance.

4.2. Comparison to Other URLB Methods

We evaluate how quickly the agent adapt to twelve downstream tasks in the same setting to URLB (Laskin et al., 2021), 2M pre-train steps and 100k fine-tune steps. Among

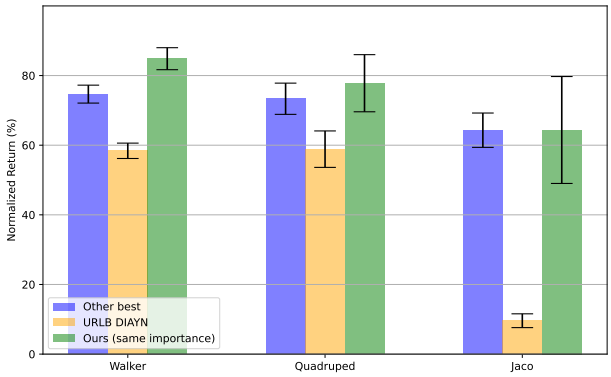

Figure 5. Results of finetuning for 100k on twelve tasks and three environments. Returns are normalized by the expert score which is trained for 2M steps with DDPG.

our proposed methods, as we have seen on the experiment above, *same importance* method is the most sample-efficient. Therefore, we finetune pretrained DIAYN with *same importance* for 3 seeds per task. We compared our method with 9 unsupervised RL algorithms including DIAYN with a vanilla fine-tuning. Expert performance is a DDPG result after 2M steps of fine-tuning (Laskin et al., 2022). *Other best* is a state-of-the-art among twelve unsupervised RL reported in URLB.

As shown in Table 2, although the simple fine-tune method without any additional weight is applied, it outperforms every other methods in 9 out of 12 tasks. Even for the tasks that other methods work better(Quadruped-stand, Jaco-reach-bottom-right, Jaco-reach-top-right), our method just slightly underperforms. Especially, it achieves an enormous improvement compared to the reported original DIAYN. From this experiment, we can recognize how important the fine-tuning method is for evaluating a skill discovery methods.

5. Conclusion

In this paper, we propose a new fine-tuning method for skill discovery for combining learned skills efficiently. By combining skills on the representation level rather than performing skills sequentially, it is possible to generalize the previous method and outperform other unsupervised RL methods in a sample efficiency manner. We emphasize the importance of fine-tuning method for measuring the performance of a skill discovery method and encourage the research direction for other skill mix methods for future work.

References

Choi, J., Sharma, A., Lee, H., Levine, S., and Gu, S. S. Variational empowerment as representation learning for goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pp. 1953–1963. PMLR, 2021.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2018.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. Urlb: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021.

Laskin, M., Liu, H., Peng, X. B., Yarats, D., Rajeswaran, A., and Abbeel, P. Cic: Contrastive intrinsic control for unsupervised skill discovery. *arXiv preprint arXiv:2202.00161*, 2022.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2016.

Liu, H. and Abbeel, P. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pp. 6736–6747. PMLR, 2021.

Salge, C., Glackin, C., and Polani, D. Empowerment–an introduction. In *Guided Self-Organization: Inception*, pp. 67–114. Springer, 2014.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Zhang, J., Yu, H., and Xu, W. Hierarchical reinforcement learning by discovering intrinsic options. *ICLR*, 2021.

A. Results

We compare our method *same importance* to *expert* which is trained by DDPG with 2M steps from (Laskin et al., 2022), *other best* which shows the best performance among 9 unsupervised RL algorithms in URLB (Laskin et al., 2021), and *URLB DIAYN*.

Table 2. Result of fine-tuning for 1×10^5 frames after pre-training for 2×10^6 frames.

DOMAIN	TASK	EXPERT	OTHER BEST	URLB DIAYN	OURS (SAME IMPORTANCE)
WALKER	FLIP	799	515±17	381±17	**658±51**
	RUN	796	439±34	242±11	**537±22**
	STAND	984	923±9	860±26	**936±11**
	WALK	971	828±29	661±26	**917±23**
QUADRUPED	JUMP	888	590±33	578±46	**645±20**
	RUN	888	465±37	415±28	**558±43**
	STAND	920	**840±33**	706±48	719±158
	WALK	866	721±56	406±64	**845±74**
JACO	REACH BOTTOM LEFT	193	134±8	17±5	**136±36**
	REACH BOTTOM RIGHT	203	**122±4**	31±4	119±39
	REACH TOP LEFT	191	124±20	11±3	**127±9**
	REACH TOP RIGHT	223	**140±7**	19±4	138±42

B. Hyperparameters

We present the best performing hyperparamter borrowed from URLB (Laskin et al., 2021).

Table 3. Hyperparameters in our experiments.

DDPG HYPERPARAMETER	VALUE
REPLAY BUFFER CAPACITY	10^6
ACTION REPEAT	1
SEED FRAMES	4000
n-STEP RETURNS	3
MINI-BATCH SIZE	24
SEED FRAMES	4000
DISCOUNT (γ)	0.99
OPTIMIZER	ADAM
LEARNING RATE	10^{-4}
AGENT UPDATE FREQUENCY	2
CRITIC TARGET EMA RATE (τ_Q)	0.01
FEATURES DIM.	1024
HIDDEN DIM.	1024
EXPLORATION STDDEV CLIP	0.3
EXPLORATION STDDEV VALUE	0.2
NUMBER PRE-TRAINING FRAMES	2×10^6
NUMBER FINE-TUNING FRAMES	1×10^5
DIAYN HYPER-PARAMETER	**VALUE**
SKILL DIM	16
SKILL SAMPLING FREQUENCY (STEPS)	50
DISCRIMINATOR NET ARCH.	$512 \rightarrow 1024 \rightarrow 1024 \rightarrow 16$ RELU MLP