Seminar Versatile Skill Learning in Robots

Andrew DeLay
Autonomous Learning Robots Lab (ALR)
Karlsruhe Institute of Technology
76131 Karlsruhe, Germany
Email: andrew.delay@student.kit.edu

***Abstract*—Recent contributions to the field of Reinforcement Learning (RL) have been concerned with the learning of *skills*, diverse behaviors of the agent that can be chosen by setting the values of latent variables which the policy is conditioned on. While some approaches omit the task reward during training, making it simply learn diverse behaviors, other approaches seek to train the agent to maximize the task reward in diverse manners. This work goes into detail on three recent approaches to diverse skill learning and examines the differences and similarities in the foundations and objectives. Finally, the experimental setups are discussed as well as which skills are manifested in the agents' behaviors.**

I. Introduction

Deep reinforcement learning (RL) algorithms have shown remarkable results in robotic control applications [1, 2, 3, 4] and games [5, 6]. While recent successes in deep RL have been largely driven by high-fidelity function approximators and increasingly sample-efficient training algorithms, a trained policy for one objective rarely adapts well to new objectives or even to small environment variations given the same agent [7]. To mitigate this, the idea of learning diverse *skills* seeks to train policies which give the agent general abilities to perform well under changing environments and agent dynamics. Learning such skills promises improved exploration in environments with sparse rewards, as well as an increased robustness of learned policies.

In this work, three recent approaches to learning diverse skills are examined: `Diversity is All You Need` by Eysenbach et al. (2018, [8]), `One Solution is Not All You Need` by Kumar et al. (2020, [9]), and `Discovering Diverse Solutions in Deep Reinforcement Learning` by Osa et al. (2021, [10]). These approaches make use of latent-conditioned policies - policies that can be adapted by setting the values of a set of *latent variables* $\boldsymbol{z}$. During training, the policy is conditioned such that the agent performs distinct styles of movement for different values of $\boldsymbol{z}$. In this context, the term *skill* refers to the latent variable $\boldsymbol{z}$ as well as the policy conditioned on $\boldsymbol{z}$. This work seeks to present the three approaches and to elaborate on consistencies and differences in the mathematical and algorithmic choices.

II. Background

All three approaches assume an RL problem under a Markov decision process (MDP) which can be represented by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \boldsymbol{\xi})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r(\boldsymbol{s}, \boldsymbol{a})$ is the reward function given an action $\boldsymbol{a}$ and state $\boldsymbol{s}$, γ is the reward discount factor, and $\boldsymbol{\xi}$ is the initial state distribution. $P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ provides the state transition probability, with $\boldsymbol{s}'$ representing the state at the time-step after $\boldsymbol{s}$. The common goal is to learn a policy which maximizes the expected sum of discounted rewards

$$R(\pi) = \mathbb{E}[R_0|\pi] \text{ where } R_t = \sum_{k=t}^{T} \gamma^{k-t} r(\boldsymbol{s}, \boldsymbol{a}). \quad (1)$$

This policy is modeled with a neural network with weights $\boldsymbol{\theta}$ and is denoted $\pi_{\boldsymbol{\theta}}$, or π for short. The papers base their algorithm for learning the policy on the soft actor-critic (SAC) algorithm [11], which is an off-policy and model-free algorithm that reuses previously collected data for optimization and includes an entropy-maximization bonus for stability and to encourage exploration. The policy π is modeled as a factored Gaussian, meaning that the neural network that represents the policy samples the action $\boldsymbol{a}$ from a learned mean and standard deviation given the network inputs $\boldsymbol{s}$ and $\boldsymbol{z}$. A noise variable ε models the stochasticity of the policy and can be fixed to zero in order to make the policy deterministic. The following definition includes the latent variable $\boldsymbol{z}$ to match the notation in the presented approaches

$$\boldsymbol{a} = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{z}) + \varepsilon \boldsymbol{\sigma}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{z}), \ \varepsilon \sim \mathcal{N}(0, 1). \quad (2)$$

Osa et al. [10] additionally implement their latent-variable conditioned algorithm based on TD3 [12], whereas [8, 9] only provide a SAC-based implementation. In order to compare the three approaches on the same grounds, the TD3-based algorithm in [10] is disregarded in the further dissemination.

III. Formulation of Mutual Information via a Variational Lower-Bound

The policy conditioned on the latent variable $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ is incentivized to have a high mutual information between the skill and the trajectories of the agent. The mutual information $\mathcal{I}(X, Y)$ between two random variables is a measure for how much knowledge of Y can be obtained by only knowing X, and vice-versa. At both extremes, the mutual information is zero if X and Y are independent, and maximal if X is a deterministic function of Y, and vice-versa. In the context of skills and trajectories, the mutual information is maximized so that the skill value $\boldsymbol{z}$ is a strong predictor of which trajectories the agent will take in the environment. Eysenbach et al. [8]

and Kumar et al. [9] seek to achieve this by maximizing the mutual information between the state and the latent variable $\mathcal{I}(\boldsymbol{s};\boldsymbol{z})$ for a given policy. A simultaneous minimization of the mutual information $\mathcal{I}(\boldsymbol{a};\boldsymbol{z}|\boldsymbol{s})$ should ensure that skills are distinguished by states and not by actions. Finally, the policy should have a high entropy for actions at a given state $\mathcal{H}(\boldsymbol{a}|\boldsymbol{s})$, a concept which is introduced by SAC [11] to capture multiple modes of near-optimal behavior. This entropy term keeps the probability density function of a SAC policy from collapsing to a single action value for a given state if multiple actions promise a high return. Together, the following objective is formulated and subsequently transformed into a sum of entropy terms using the relationship $\mathcal{I}(Y;X) = \mathcal{I}(X;Y) = \mathcal{H}(X) - \mathcal{H}(X|Y)$ and the definition of entropy $\mathcal{H}(X) = -\mathbb{E}[\log p(X)]$

$$\begin{aligned}
&\mathcal{I}(\boldsymbol{s};\ \boldsymbol{z}) + \mathcal{H}(\boldsymbol{a}|\boldsymbol{s}) - \mathcal{I}(\boldsymbol{a};\boldsymbol{z}|\boldsymbol{s}) \\
&= (\mathcal{H}(\boldsymbol{z}) - \mathcal{H}(\boldsymbol{z}|\boldsymbol{s})) + \mathcal{H}(\boldsymbol{a}|\boldsymbol{s}) - (\mathcal{H}(\boldsymbol{a}|\boldsymbol{s}) - \mathcal{H}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})) \\
&= \mathcal{H}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z}) - \mathcal{H}(\boldsymbol{z}|\boldsymbol{s}) + \mathcal{H}(\boldsymbol{z}) \\
&= \mathcal{H}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z}) + \mathbb{E}_{(\boldsymbol{s},\boldsymbol{z})\sim p_\pi}[\log p(\boldsymbol{z}|\boldsymbol{s})] + \mathcal{H}(\boldsymbol{z}). \qquad (3)
\end{aligned}$$

Here, Eysenbach et al. [8] state that the distribution $p(\boldsymbol{z}|\boldsymbol{s})$ cannot be computed directly and instead use Jensen's inequality to show that approximating $p(\boldsymbol{z}|\boldsymbol{s})$ with a learned discriminator $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s})$ yields a lower bound to the objective, called a *variational lower bound*. This discriminator is represented by a neural network with weights $\boldsymbol{\phi}$. As their approach is based on SAC [11], there is already a mechanism in place that maximizes the entropy of the policy w.r.t. the actions, so that the term $\mathcal{H}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z})$ is removed from the objective

$$\begin{aligned}
(3) &= \mathcal{H}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z}) + \mathbb{E}_{(\boldsymbol{s},\boldsymbol{z})\sim p_\pi}[\log p(\boldsymbol{z}|\boldsymbol{s})] + \mathcal{H}(\boldsymbol{z}) \\
&= \mathcal{H}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z}) + \underbrace{\mathbb{E}_{\boldsymbol{s}\sim\beta(\boldsymbol{s})}[D_{\mathrm{KL}}(p(\boldsymbol{z}|\boldsymbol{s})\|q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}))]}_{\geq 0} \\
&\quad + \mathbb{E}_{(\boldsymbol{s},\boldsymbol{z})\sim p_\pi}[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s})] + \mathcal{H}(\boldsymbol{z}) \\
&\geq \mathcal{H}(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{z}) + \mathbb{E}_{(\boldsymbol{s},\boldsymbol{z})\sim p_\pi}[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s})] + \mathcal{H}(\boldsymbol{z}) \\
&\geq \mathbb{E}_{(\boldsymbol{s},\boldsymbol{z})\sim p_\pi}[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s})] + \mathcal{H}(\boldsymbol{z}). \qquad (4)
\end{aligned}$$

Osa et al. [10], on the other hand, maximize $\mathcal{I}(\boldsymbol{s},\boldsymbol{a};\boldsymbol{z})$, thus including the action into the term in order to "encode the diversity of actions for a specified state into the latent variable" [10, Sec. 4.1]. Here as well, the term for the mutual information is split up and the distribution $p(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$ is approximated with a learned discriminator $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$ which lower-bounds the objective in the same way (compare Eq. (4)), motivated in the inability to compute $p(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})$ directly

$$\begin{aligned}
\mathcal{I}(\boldsymbol{s},\boldsymbol{a};\boldsymbol{z}) &= \mathcal{H}(\boldsymbol{z}) - \mathcal{H}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a}) \\
&= \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\sim p_\pi}[\log p(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})] + \mathcal{H}(\boldsymbol{z}) \\
&\geq \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\sim p_\pi}[\log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a})] + \mathcal{H}(\boldsymbol{z}). \qquad (5)
\end{aligned}$$

Despite starting out with very different objectives and motivations for maximizing the mutual information, the approaches in Equations (4) and (5) only differ in the arguments to the discriminator $q_{\boldsymbol{\phi}}$. In addition, all three papers maximize the skill entropy $\mathcal{H}(\boldsymbol{z})$ by setting $p(\boldsymbol{z})$ as a uniform distribution, meaning that $\boldsymbol{z}$ is sampled from a uniform distribution at the beginning of each training episode. Being based on the off-policy SAC algorithm, previously collected data from the replay buffer $\mathcal{B}$ can be used for maximizing the objectives.

Regarding the latent variable vector, [8, 9] confine $\boldsymbol{z}$ to discrete values, whereas [10] uses both continuous and discrete latent variables. While [8, 9] don't elaborate on their decision, Osa et al. [10] draw connections to infoGAN [13] where continuous latent variables can be used to adjust the output of written digits in a continuous manner, allowing fine adjustments to the rotation and the line width.

IV. Objectives to Facilitate Diverse Skill Learning

Building on the very similar objectives of mutual information maximization, the three papers diverge in their formulation of the optimization problems. Eysenbach et al. [8] emphasize their work on learning skills without a reward function. Here, the reward given by the environment is replaced with a pseudo-reward which is formulated from the variational lower bound in Eq. (4), yielding

$$\tilde{r}(\boldsymbol{s},\boldsymbol{a}) = \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}) + \mathcal{H}(\boldsymbol{z}). \qquad (6)$$

Maximizing this pseudo-reward causes the agent to visit different states for different values of the discrete latent variables, resulting in movements which are not directed to solve a specific task, as no task reward is given during training. This can result in an agent which has no useful skills, i.e. doesn't achieve a substantial task reward at test-time. For agents whose movements are restricted to a plane, which is the case in the Hopper or Half-Cheetah environments, the randomly learned skills include useful locomotive abilities, such as moving forward, backward or in-place. The less restricted the environment becomes though, the less likely it is for the learned skills to perform well given a task reward. Eysenbach et al. [8] observe this in the Ant environment where the agent fails to move in straight lines. Here, the agent would likely not achieve a high task reward when the goal is to move quickly in a certain direction.

Incorporating the same pseudo-reward (6) into their optimization objective, Kumar et al. [9] seek to train the agent so that it only possesses skills which also achieve a near-optimal total task reward $R(\pi_{\boldsymbol{\theta}}) \geq R(\pi^*) - \varepsilon$ so that each skill is useful. Knowing the near-optimal total task reward $R(\pi^*)$ requires training a non-latent-variable conditioned agent on the task beforehand. This substantially increases the computational effort needed for this approach, as two agents need to be trained in total. Kumar et al. [9] call this approach the *Structured Maximum Entropy RL* (SMERL) algorithm, which maximizes the objective in Eq. (1) using the following reward, with $\alpha > 0$, $\varepsilon > 0$, and the indicator function $\mathbb{I}$

$$r_{\mathrm{SMERL}}(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \alpha \mathbb{I}_{R(\pi_{\boldsymbol{\theta}})\geq R(\pi^*)-\varepsilon}\tilde{r}(\boldsymbol{s},\boldsymbol{a}).$$

While this approach guarantees that the agent learns to solve the given task well in many different ways, the agent

will not adapt well to new tasks, as there is less diversity in the skills. To provide an example: If a Half-Cheetah is trained to move forward using SMERL, it will likely learn to do so in different locomotive styles. If the reward changes to encourage backwards movement however, the agent will probably fail to do so, as all of the skills have been conditioned on forward movement.

Diverging from the previous approaches, Osa et al. [10] don't incorporate the variational lower bound of the mutual information as unsupervised rewards. Instead, they maximize the objective (5) directly using back-propagation. This is achieved by defining an objective $\mathcal{J}_{\text{info}}$ which is only optimized every d_{info} time steps

$$\mathcal{J}_{\text{info}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{z})\sim\mathcal{B}}[\tilde{W} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{z}))].$$

Where W and $\tilde{W}$ are the un-clipped and clipped *importance weights*, respectively

$$W = \frac{\exp(\min_{i=1,2} Q_i(\boldsymbol{s}, \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{z}), \boldsymbol{z}))}{\sum_{\bar{\boldsymbol{s}},\bar{\boldsymbol{a}},\bar{\boldsymbol{z}}\in\mathcal{B}} \min_{i=1,2} Q_i(\bar{\boldsymbol{s}}, \bar{\boldsymbol{a}}, \bar{\boldsymbol{z}})}, \tag{7}$$

$$\tilde{W} = \max(1 - c_{\text{clip}}, \min(W, 1 + c_{\text{clip}})).$$

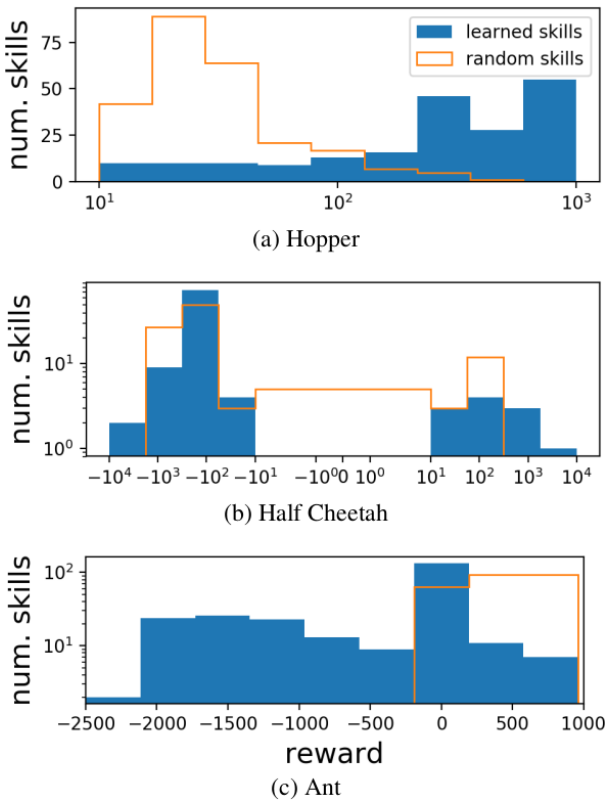
To compute $\mathcal{J}_{\text{info}}$, the log likelihood of the discriminator is calculated among a set of state-skill pairs which are sampled from the replay-buffer $\mathcal{B}$. For the corresponding action, the mean of the Gaussian policy $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{z})$ in Eq. (2) is calculated so that a gradient can be calculated w.r.t. the weights of the policy $\pi_{\boldsymbol{\theta}}$. The policy itself is optimized every d_{atr} time-steps according to the SAC loss-function for the actor. In the experiments, Osa et al. [10] set $d_{\text{atr}} = 2$ and $d_{\text{info}} = 3$ and claim that reducing d_{info} results in more diverse behaviors at the cost of a lower average reward. Thus, the hyperparameters d_{info} and d_{atr} allow for a fine-tuning of this trade-off. For $d_{\text{info}} \to \infty$ and $d_{\text{atr}} = 1$, the algorithm resembles SAC [11] and no diverse skills are learned. Because the information term isn't included in the reward, [10] redefines the training objective for the policy in Eq. (1) as

$$\max_{\pi,\boldsymbol{\theta}}(\mathbb{E}[R|\pi] + \mathcal{J}_{\text{info}}(\pi, \boldsymbol{\theta})).$$

The importance weight in Equation (7) is reminiscent of a softmax function and normalizes the expected return of the current state, policy-mean and skill tuple $(\boldsymbol{s}, \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{z}), \boldsymbol{z})$. For those tuples with a comparatively high expected return, the value of the discriminator will have a more profound effect on the gradient of the weights $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ when optimizing the mutual information. This encourages the policy to learn diverse skills that perform actions that promise high returns. Because of this, the learned skills aim to achieve high rewards which limits the agent's ability to achieve high rewards when the task changes, as discussed above in the case of SMERL [9].

V. Experiments

The three algorithms are evaluated in the OpenAI Gym [14] environments which use the MuJoCo physics engine [15]. Eysenbach et al. [8] apply their algorithm to the Hopper, Half-Cheetah and Ant environments in order to compare the



(a) Hopper

(b) Half Cheetah

(c) Ant

Fig. 1. Rewards of the trained, latent-variable conditioned policy compared with a random policy [8, Fig. 6]. The agent receives no task reward during training.

achieved rewards of the different skills to a random policy. Their policies, which were trained without observance of the environment reward, achieve a wide range of different accumulated rewards across all skills. The achieved-rewards histogram shown in Fig. 1 is distinct from that of a random policy which shows that diverse skills are learned. The histogram shows that the less constrained the movement of the agent is, the fewer skills achieve a high task reward in a given environment: While many Hopper skills reach a substantially high reward, only relatively few Ant skills do.

Kumar et al. [9] evaluate their policies on the Half-Cheetah, Hopper, and Walker2d environments with a modified task reward function. In the Half-Cheetah environment, the reward is maximized when the agent reaches a certain target location. In the Hopper and Walker2d environments, the reward encourages the agent to move at a specific speed. Osa et al. [10] reward a specific, constant speed as well in the Hopper, Walker2d, and Humanoid environments which they evaluate their policy on. In the rewards of the unmodified Gym environments, the term for velocity is not capped in order to encourage high speeds. The reason why [9, 10] cap the velocity reward is likely due to the indifference of skills which achieve maximal velocity. A comparison can be made with human sports competitions: In sprinting disciplines, the movement styles of all competitors only show minute variations. RL agents which are trained to achieve a velocity as high as possible will have skills that are not easy to

distinguish. In order to obtain visibly different locomotive styles the velocity reward must thus be capped.

The further discussion of the experiments concerns the *few-shot robustness*, a protocol introduced by Kumar et al. [9] which evaluates the robustness of the learned policy to perturbations of the agent dynamics and the environment. The protocol works as follows: A policy is trained in a given environment, modeled by the MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \boldsymbol{\mu})$. Then, in order to evaluate the few-shot robustness, the agent is placed into a new MDP $\mathcal{M}' = (\mathcal{S}, \mathcal{A}, P', r', \gamma, \boldsymbol{\mu})$ with identical state and action spaces as $\mathcal{M}$. In this new MDP, the agent is confronted with perturbations and is allowed only a handful of training episodes k to adapt to the new circumstances. In [9], the agent should achieve a high reward with obstacles in its way, external forces applied to it joints, as well as joint failures. Osa et al. [10] evaluate their algorithms using the few-shot robustness protocol, but only in the Walker2d environment and with different perturbations such as changing the length of a single leg or moving the position of the knee up or down the leg. The few-shot robustness is compared with *SMERL* [9] given the same circumstances and an outperformance of Osa et al.'s algorithm [10] is asserted.

But how are the skills manifested in the behavior of the agents? This question is evaluated on the results of the three approaches in the Hopper environment, the only environment the three papers have in common. Describing the effects of the different values of the latent variables on the motion of the agent is impossible for Kumar et al. [9] though, as no images or videos of any agent's motion is provided. In [8, 10], the papers provide snapshots of the agents at different time steps of its motion, as well as videos. The Hopper agent by Eysenbach et al. [8] shows very distinct movement patterns, as the policy was not exposed to the environment reward during training. Thus, changing the value of the categorical latent variable z results in a forward or backward movement at different speeds, although the agent fails to keep its balance for longer than a few steps. The Hopper agent by Osa et al. [10] was explicitly trained to achieve a high task reward which results in all the skills showing a forward movement. Varying the continuous or discrete latent variables causes the Hopper agent to keep the upper/lower knee straight or bent throughout the episode, or influences the angle of the foot, resulting in a tip-toeing of the agent in some cases.

VI. Related Work

Related work by Sharma et al. [16] combines the model-free approach of learning diverse skills as in Eysenbach et al. [8] with a model-based approach. It harnesses the latent variables and seeks to learn the skill-transition dynamics $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{z})$ of the environment, in contrast to model-based RL algorithms which learn the action-transition dynamics $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$. A model-predictive control planner then uses the learned skill-transition dynamics to generate a trajectory of skills $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$ in order to achieve a high task reward. This combination of model-free and model-based RL allows for a single latent-variable conditioned policy to solve multiple tasks, while the model-predictive control algorithm only needs to plan in the low dimensional skill space instead of the higher dimensional action space.

VII. Conclusion

The three discussed approaches [8, 9, 10] train a latent-variable conditioned policy which allows the agent to adapt different locomotive styles. The theoretical foundation of maximizing the mutual information between the latent variable and the state, or state-action pair, is very similar among the approaches. The algorithmic basis, soft actor-critic [11], is a shared feature as well. The greatest difference between the three papers lies in the formulation of the maximization objectives which facilitate the diverse skill learning. A high-level comparison is summarized in Table I. While the policies learned in Eysenbach et al. [8] show the most diverse skills as they aren't directed to solve a specific task, the agent is short-lived as no incentive is given for survival and fails to learn the key useful skill of balance. The skills learned in [9, 10] achieve higher task rewards simply because they are optimized to do so, but still yield many skills which end the episode quickly.

The skills learned in the three approaches aren't always useful, as the methods either don't incentivize the performance in the environment at all or only define a narrow incentive,

	`Diversity is All You Need`, Eysenbach et al. [8]	`One Solution is Not All You Need`, Kumar et al. [9]	`Discovering Diverse Solutions in Deep Reinforcement Learning`, Osa et al. [10]
Algorithm base	SAC [11]		SAC [11] and TD3 [12]
Skill vector $\boldsymbol{z}$	Discrete values only		Discrete and continuous values
Mutual information formulation	Between the skill and the state, $\mathcal{I}(\boldsymbol{s}; \boldsymbol{z})$		Between the skill and the state-action pair, $\mathcal{I}(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{z})$
Maximization objective	$\max \mathbb{E}[R\|\pi]$		$\max(\mathbb{E}[R\|\pi] + \mathcal{J}_{\text{info}}(\pi, \boldsymbol{\theta}))$
Reward	Only pseudo-reward $\tilde{r}(s, a)$ (6)	Environment reward $r(s, a)$ and pseudo-reward $\tilde{r}(s, a)$ (6)	Only environment reward $r(s, a)$
Evaluated environments	Half Cheetah, Ant, Hopper	Half Cheetah, Walker2d, Hopper	Humanoid, Walker2d, Hopper

TABLE I
A HIGH-LEVEL COMPARISON OF THE THREE DISCUSSED APPROACHES

such as moving forward at a specific speed. In Kumar et al. [9] a method of testing the robustness of a learned policy is introduced, called *few-shot robustness*, which is also picked up by Osa et al. [10]. While this method is useful for testing robustness, it is a contribution to the field of RL in general and is not tied to skill learning. Kumar et al. [9] raise the issue that it is unknown how many different skills are necessary for any given task, and assume that continuous latent variables may be the key to the answer. Osa et al. [10] implement continuous latent variables, but the question remains unanswered. A further open question is that it can't be predicted what the latent-variable vector encodes. The diversity of the skills can only be influenced via the hyperparameters, although these aren't an absolute measure for diversity, and finding the right balance between diversity and achieved reward requires much trial-and-error.

References

[1] Sergey Levine et al. "End-to-End Training of Deep Visuomotor Policies". In: *Journal of Machine Learning Research* 17.39 (2016), pp. 1–40. URL: http://jmlr.org/papers/v17/15-522.html.

[2] Xue Bin Peng et al. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (May 2018), pp. 3803–3810. DOI: 10.1109/ICRA.2018.8460528. arXiv: 1710.06537. URL: http://arxiv.org/abs/1710.06537 (visited on 12/12/2020).

[3] OpenAI et al. "Solving Rubik's Cube with a Robot Hand". In: *arXiv:1910.07113 [cs, stat]* (Oct. 15, 2019). arXiv: 1910.07113. URL: http://arxiv.org/abs/1910.07113 (visited on 12/12/2020).

[4] Shixiang Gu et al. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017, pp. 3389–3396. ISBN: 978-1-5090-4633-1. DOI: 10.1109/ICRA.2017.7989385. URL: http://ieeexplore.ieee.org/document/7989385/ (visited on 06/22/2021).

[5] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature14236. URL: http://www.nature.com/articles/nature14236 (visited on 12/12/2020).

[6] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (Jan. 28, 2016), pp. 484–489. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature16961. URL: http://www.nature.com/articles/nature16961 (visited on 06/22/2021).

[7] Chiyuan Zhang et al. "A Study on Overfitting in Deep Reinforcement Learning". In: *arXiv:1804.06893 [cs, stat]* (Apr. 20, 2018). arXiv: 1804.06893. URL: http://arxiv.org/abs/1804.06893 (visited on 06/22/2021).

[8] Benjamin Eysenbach et al. "Diversity is All You Need: Learning Skills without a Reward Function". In: *arXiv:1802.06070 [cs]* (Oct. 9, 2018). arXiv: 1802.06070. URL: http://arxiv.org/abs/1802.06070 (visited on 04/19/2021).

[9] Saurabh Kumar et al. "One Solution is Not All You Need: Few-Shot Extrapolation via Structured MaxEnt RL". In: *Advances in Neural Information Processing Systems* 33 (2020).

[10] Takayuki Osa, Voot Tangkaratt, and Masashi Sugiyama. "Discovering Diverse Solutions in Deep Reinforcement Learning". In: *arXiv:2103.07084 [cs, stat]* (Mar. 11, 2021). arXiv: 2103.07084. URL: http://arxiv.org/abs/2103.07084 (visited on 04/19/2021).

[11] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[12] Scott Fujimoto, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1587–1596.

[13] Xi Chen et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *arXiv:1606.03657 [cs, stat]* (June 11, 2016). arXiv: 1606.03657. URL: http://arxiv.org/abs/1606.03657 (visited on 04/30/2021).

[14] "OpenAI Gym". In: (). URL: http://arxiv.org/abs/1606.01540.

[15] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012, pp. 5026–5033. ISBN: 978-1-4673-1736-8. DOI: 10.1109/IROS.2012.6386109. URL: http://ieeexplore.ieee.org/document/6386109/ (visited on 07/14/2021).

[16] Archit Sharma et al. "Dynamics-Aware Unsupervised Discovery of Skills". en. In: *arXiv:1907.01657 [cs, stat]* (Feb. 2020). arXiv: 1907.01657. URL: http://arxiv.org/abs/1907.01657 (visited on 07/14/2021).