Multi-Goal Reinforcement Learning with Conditional Variational Autoencoders

Maximilian Sieb, Hariank Muthakana, Satyaki Chakraborty *

1 Introduction

Multi-goal and multi-task reinforcement learning are interesting topics that have gotten a lot of attention lately. Given an environment and a goal, how do we transfer knowledge and policies to another goal or another task? In this work, we investigate multi-goal/multi-task learning using variational autoencoders (VAEs). We work in the MuJoCo environment, which has high-dimensional state-space representation as well as complex goals and tasks.

We will give an overview of related approaches and environments. Then, we will formally introduce the definitions of goals and tasks in the field of reinforcement learning, followed by an extensive description of our proposed method to tackle multi-task and multi-goal learning effectively. More specifically, we will augment off-policy RL algorithms with conditional variational autoencoders (CVAEs) to learn a policy representation that can capture the different modalities of a multi-goal task, possibly extending it across different tasks as well. Finally, we compare the knowledge transfer with and without the added autoencoder.

2 Background

2.1 Hindsight Experience Replay

Hindsight Experience Replay (HER) [1] is a method that allows off-policy RL algorithms to learn from sparse rewards in a sample-efficient way. The key idea is to learn from failed episodes by treating the final state reached as a new "goal". The trajectory and state reached are added into the replay buffer and later when we replay, an algorithm like DDPG [7] can learn from trajectories with a variety of goals. With HER, the reward function does not need to be manually shaped with domain knowledge of the environment, and can simply be a binary signal of whether the goal was reached or not. Because of how it encourages generalization, this technique lends itself well to multi-goal learning and even improves single-goal performance.

2.2 Multi-Task & Multi-Goal RL

For a fruitful discussion and investigation of multi-goal and multi-task reinforcement learning, the terms *goal* and *task* have to be appropriately defined. In this work, we define a goal g as a mapping from states to $\{0, 1\}$, indicating whether the goal has been reached or not. This definition is analogous to the definition laid out in [1]. A goal can be more practically defined as a different goal setting within the same dynamical environment, i.e. the state-space representation and its dimension as well as all other domain-specific parameters are equal except for explicit goal-dependent rewards and termination of an episode. Multi-goal reinforcement learning can be formalized by introducing another parameter g into the system, thus making the policy not only state, but also goal dependent. Therefore, we learn one unified stochastic policy $\pi(s, g)$ to account for different goals instead of learning one policy for each goal separately [12].

*msieb@cs.cmu.edu, hmuthaka@cs.cmu.edu, schakra1@cs.cmu.edu

(2016).

As a task ω, we define in this work a setting that is "similar" in the sense that the actor has the same capabilities across all tasks, but the environment dynamics and the interactions with the environment are different. For example, consider a robotic hand trying to hold a die and turn it such that the side with the number six points upwards. Another *goal* would be to turn it such that the number three faces upwards, whereas a different *task* would be to have a spherical object with a set of number printed on top of it and turning it such that a number faces upwards.Multi-task reinforcement can be formalized by using the same reasoning as for multi-goal RL. Generally policies $\pi_{\omega(s,g)}$ are learned for each task separately, or unified into a single policy $\pi(s, g)$. This process of unifying multiple policies into one unifying is known as policy distillation [17, 11].

3 Methods

We will investigate the use of Variational Autoencoders (VAEs) [6] to encode stochastic policies conditioned on other inputs. This Conditional Variational Autoencoder (CVAE) [15] would then output an action distribution conditioned on different goals and states in analogy to learning separate policy networks for each goal. We can learn goal-dependent CVAEs and fuse them (either in parallel during the individual learning process or after finishing the independent learning) to one unified CVAE that is conditioned on the current goal and state to output an action distribution for that state and goal. Alternatively, we could directly maintain one single CVAE model that consistently incorporates more training data across different runs with different goal settings. In this work, we will investigate the second case, since maintaining one single model allows for learning a unified representation straight from the data instead of adding another complexity layer to fuse independent CVAEs. For an overview of the proposed architecture, refer to figure 1.

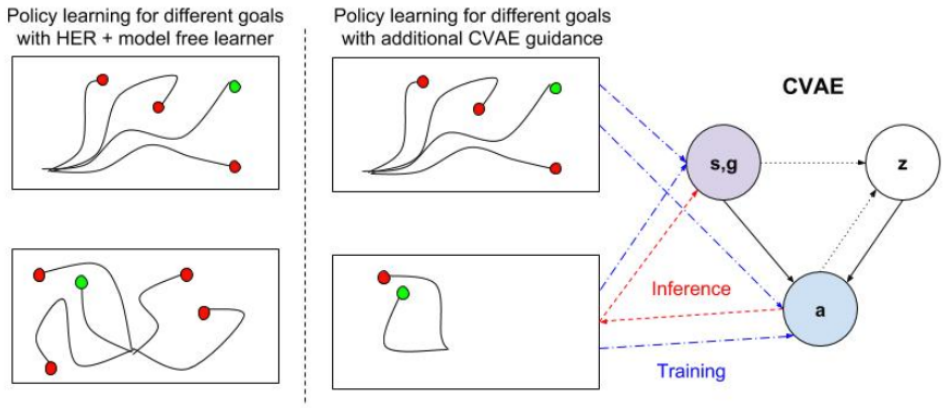

Figure 1: Policy learning with CVAE guidance

Given several goal-dependent policies $\pi_g(a|s)$, we formally define the "true" action distribution of the current task as $p(a)$. This is analogous to the "evidence" in other settings. We extend this model by considering a Conditional Variational Autoencoder (CVAE) to be able to condition the action distribution on a given input, which in our case would be the states and/or goal. Overall, we are looking to approximate the distribution $p(a|(s, g))$. Introducing the latent variable z, we have an encoder $q_\theta(z|a, (s, g))$ and a decoder $p_\varphi(a|z, (s, g))$ and we can define the CVAE reconstruction loss like in [15] as

$$\mathcal{L}(\theta, \varphi) = -\mathbb{E}_{z\sim q_\theta(z|a,(s,g))}[\log p_\varphi(a|z, (s, g))] + KL(q_\theta(z|a, (s, g)||p_\varphi(z|(s, g)))$$

where $(a, s) \sim \tau$ and τ is the currently evaluated trajectory produced by the policy learner network.

How do we generate data for the CVAE? Overall, the goal is to model the underlying data distribution of the actions relevant to the task at hand. For that reason, we run any (continuous) policy learning algorithm such as DDPG on the task and use the action data generated through the policy learner to train the CVAE. Ideally, we only take samples from successful runs in order to learn an action

distribution that is representative of "successful" actions and not just random actions, analogous to feeding "real" images into a VAE in an image setting and not just noise. In a one-goal setting, one would use a variety of different runs and setups that solve a task successfully and then train the CVAE with that data.

In a sparse-reward environment, we want to see how using Hindsight Experience Replay can help the CVAE learn more efficiently by introducing substitute goals to additionally get valuable training data from non-successful runs that are in itself "successful" with respect to the substitute goal. To do so, we not only condition the VAE on the state, but also on the current goal. For the current substitute goal, we just learned a successful policy and we can train the CVAE on that one. The hope is that after a certain amount of training, the CVAE will learn a latent representation that can aid the policy learner towards reaching the actual goal quicker by providing guidance on the variance and value of actions in the current state and given the current goal, trained on experience from previous runs. Assuming the CVAE is trained on a couple of different goals, we can now start to enhance the main policy learner (e.g. DDPG + HER). To do so, we generate enough samples from the conditional distribution $p_\theta(a|s, g)$ to calculate the sample variance. We can now enhance the local exploration rate of the learner; if the CVAE predicts high variance at that state, the learner should explore more and vice versa. This yields an enhanced exploration tactic instead of solely being based on, for example, a simple decaying ϵ-greedy policy.

A more structured approach would include to impose an additional loss term on the policy learner: If we are using a stochastic policy such as TRPO, we can enforce an additional loss term via

$$\mathcal{L}_{sim}(\theta_{actor}) = KL(\pi_g(a|s)||p_\theta(a|(s, g)))$$

to guide the action selection of the learner to be more similar to the distribution of the CVAE at that state (and goal).

Even in a setup where we use non-stochastic policies such as DDPG, we can usually treat them as stochastic policies since they are often implementing some kind of exploration noise on top of the deterministic output. Defining the explorative policy as $\tilde{\pi}_g(a|s) = \mu_g(s) + \epsilon$, where $\mu_g(s)$ is the deterministic goal-dependent state-to-action mapping of any deterministic policy learner and ϵ is any stochastic noise distribution, we can introduce the same loss as defined before via

$$\mathcal{L}_{sim}(\theta_{actor}) = KL(\tilde{\pi}_g(a|s)||p_\theta(a|(s, g)))$$

For any policy learning algorithm, the combined loss of the actor-network would then be a combination of the original actor loss and the CVAE imposed loss:

$$\mathcal{L}_{total}(\theta_{actor}) = \mathcal{L}_{original} + \alpha\mathcal{L}_{sim}$$

Keeping in mind that the analysis so far was restricted to sparse environment or dense environments with an implicit notion of a goal to evaluate "success", we also want to investigate how we can use Importance Weighted Autoencoders (IWAE) to incorporate reward feedback in dense-reward environments to train the CVAE through continous reward feedback regardless of the actual episodic success.

If time permits, the next investigative step would be to train multiple one-task goal-conditioned CVAEs for separate tasks ω and then fuse them to one unified task- and goal-dependent CVAE or an ensemble of CVAEs. Since the representation/dimensionality of the goal is dependent on the task now, we require an intermediate feature transformation.

4 Dataset

We will mainly evaluate our approach using the *Robotics* environment from OpenAI that implements the MuJoCo physics simulator. More specifically, we will evaluate on the *Fetch* and the *ShadowHand* environment. Regarding the metrics to be evaluated, we will compare how long it takes the agent to successfully reach the specified goal configuration in the given environment. We will then investigate if the learner improves if given a different goal if it receives additional guidance from the trained CVAE compared to the vanilla implementation of HER and a model-free policy learning method.

As mentioned above, if time permits, we will further investigate if the CVAE helps across environments that are similar, for example trying to manipulate an egg after learning how to manipulate a cube with the ShadowHand.

5 Preliminary Results

We first evaluate baseline methods for multi-goal learning, namely HER + DDPG, trained on the *FetchReach* and *FetchPickAndPlace* environments with sparse rewards and randomized goal setting for each run. For each environment, we compare the success rate when training from a randomly initialized policy versus training from the best policy found from all previous runs (for different goals). In this case, we trained policies on 5 different goal settings, always keeping the best policy that has been trained throughout the entire experiment. For the final results, we ran the same setting 3 times and then report the average results to account for random seeds. The results are summarized in Figures 2-5.

Generally, it is evident that "pretraining" on different goals does not seem to aid performance in a multi-goal setting. That is exactly what we hope can be improved upon by using a CVAE to transfer "knowledge" across multiple goals more effectively. Nonetheless, we are going to conduct more runs to pretrain the vanilla HER + DDPG policies more extensively on multiple goals to see if more longer-term training would induce some kind of transfer learning.

Additionally, we plan on conducting more baseline experiments with the *HandManipulateBlock* and *HandManipulateEgg* enviroments - however, [1] used 19 CPUs to parallelize the training for these environments to reach an adequately high success rate after 200 epochs. Until now, we did not have the computational resources to conduct these types of experiments.

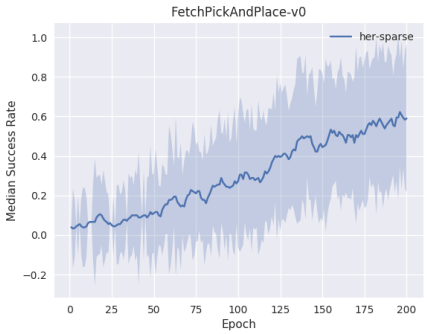

Figure 2: FetchPickAndPlace, HER+DDPG, random initialization

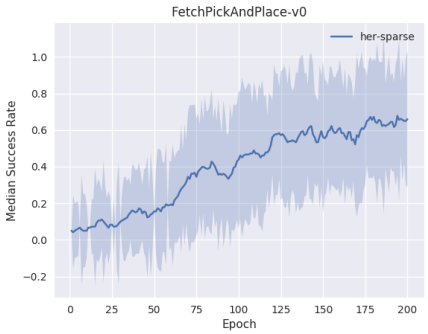

Figure 3: FetchPickAndPlace, HER+DDPG, best previous policy

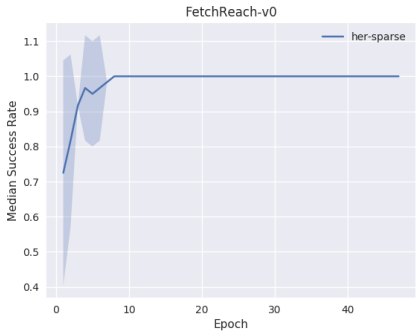

Figure 4: FetchReach, HER+DDPG, random initialization

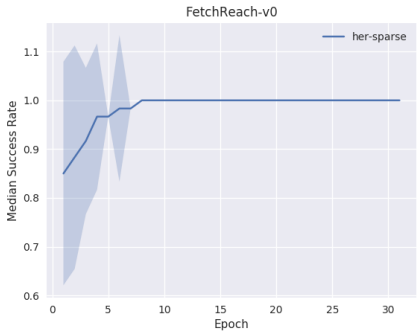

Figure 5: FetchReach, HER+DDPG, best previous policy

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. 2017.

[2] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36, 2012.

[3] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-Conquer Reinforcement Learning. pages 1–11, 2017.

[4] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning that Matters. 2017.

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[6] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. (Ml):1–14, 2013.

[7] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. 2015.

[8] David Pardoe and Peter Stone. Boosting for regression transfer. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 863–870. Omnipress, 2010.

[9] Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Actor-Mimic Deep Multitask and Transfer Reinforcement Learning. pages 1–16, 2016.

[10] Paulo Rauber, Filipe Mutz, and Juergen Schmidhuber. Hindsight policy gradients. 2017.

[11] Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy Distillation. pages 1–13, 2015.

[12] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal Value Function Approximators. *Proceedings of The 32nd International Conference on Machine Learning*, pages 1312–1320, 2015.

[13] John Schulman, Xi Chen, and Pieter Abbeel. Equivalence Between Policy Gradients and Soft Q-Learning. pages 1–15, 2017.

[14] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 387–395, 2014.

[15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning Structured Output Representation using Deep Conditional Generative Models. *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

[16] Matthew E Taylor and Peter Stone. An introduction to intertask transfer for reinforcement learning. *Ai Magazine*, 32(1):15, 2011.

[17] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust Multitask Reinforcement Learning. 2017.

[18] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.