ProtoGE: Prototype Goal Encodings for Multi-goal Reinforcement Learning

Silviu Pitis*
University of Toronto, Vector Institute
Toronto, ON, Canada
spitis@cs.toronto.edu

Harris Chan*
University of Toronto, Vector Institute
Toronto, ON, Canada
hchan@cs.toronto.edu

Jimmy Ba
University of Toronto, Vector Institute
Toronto, ON, Canada
jba@cs.toronto.edu

Abstract

Current approaches to multi-goal reinforcement learning train the agent directly on the desired goal space. When goals are sparse, binary and coarsely defined, with each goal representing a set of states, this has at least two downsides. First, transitions between different goals may be sparse, making it difficult for the agent to obtain useful control signals, even using Hindsight Experience Replay [1]. Second, having trained only on the desired goal representation, it is difficult to transfer learning to other goal spaces.

We propose the following simple idea: instead of training on the desired *coarse* goal space, substitute it with a *finer*—more specific—goal space, perhaps even the agent's state space (the "state-goal" space), and use Prototype Goal Encodings ("ProtoGE") to encode coarse goals as fine ones. This has several advantages. First, an agent trained on an appropriately fine goal space receives more descriptive control signals and can learn to accomplish goals in its desired goal space significantly faster. Second, finer goal representations are more flexible and allow for efficient transfer. The state-goal representation in particular, is *universal*: an agent trained on the state-goal space can potentially adapt to arbitrary goals, so long as a Protoge map is available. We provide empirical evidence for the above claims and establish a new state-of-the-art in standard multi-goal MuJoCo environments.

Keywords: multi-goal reinforcement learning, task specification, transfer learning, hindsight experience replay

1 Introduction, Background & Related Work

Humans can often accomplish specific goals more readily that general ones. Although more specific goals are, by definition, more challenging to accomplish than more general goals, evidence from management and educational sciences supports the idea that "specific, challenging goals lead to higher performance than easy goals" [9]. We find evidence of this same effect for reinforcement learning (RL) agents in multi-goal environments. Our work establishes a new state-of-the-art in standard multi-goal MuJoCo environments and suggests several novel research directions.

Multi-Goal Reinforcement Learning We consider the multi-goal RL setting, where an agent interacts with an environment and learns to accomplish different goals. The problem is described by a generalized Markov Decision Process (MDP) $\langle S, A, T, G \rangle$, where S, A, T, and G are the state space, action space, transition function and goal space, respectively [15, 16]. In the most general version of this problem each goal is a tuple $g = \langle R_g, \gamma_g \rangle$, where $R_g : S \to \mathbb{R}$ is a reward function and $\gamma_g \in [0, 1]$ is a discount factor [16], so that "solving" goal $g \in G$ amounts to finding an optimal policy in the classical MDP $\langle S, A, T, R_g, \gamma_g \rangle$. We focus on the sparse, binary reward case where each goal g corresponds to a set of "success" states, $\mathcal{S}(g)$, with $R_g : S \to \{-1, 0\}$ and $R_g(s) = 0$ if and only if $s \in \mathcal{S}(g)$ [13]. In this setting, the agent must learn to achieve and maintain success.

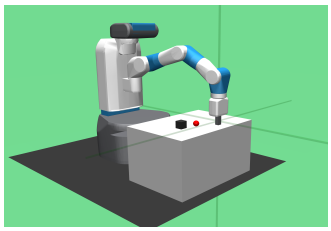

Fig. 1: In `Push`, the agent must push the black box onto the red target.

We use three multi-goal `Fetch` environments from OpenAIGym [3]: `Push`, `PickAndPlace`, and `Slide` (`v0`) [13]. In `Push` and `PickAndPlace`, the agent must use its gripper to move a box to the desired location (Figure 1). In `Slide`, the agent must hit a puck so it slides onto and stops in the desired location beyond the reach of the robot. A discount factor of $\gamma = 0.98$ is used and the environments reset (a new goal is sampled) every 50 steps. Goals are specified by a 3-dimensional vector of x, y and z coordinates. The goal is satisfied and reward of 0 is obtained so long as the box or puck is within an epsilon ball of the goal. An episode is a "success" only if the goal is satisfied on the final (50th) step.

Goal-Conditioned Actor-Critic Algorithms Many RL algorithms can be decomposed into actor (policy) and critic (value function). The DQN algorithm, for discrete action spaces, parameterizes both a greedy actor and a critic using the same deep neural network [10]. DDPG [7], the continuous action space equivalent of DQN, parameterizes actor and critic separately. Both DQN and DDPG are off-policy algorithms and use a replay buffer to store past experiences; this buffer is sampled from to train the actor and critic networks. To use DQN and DDPG in the multi-goal setting, we adopt the standard approach, which generalizes the actors (critics) to be functions of not only the state s (and action a), but also the goal g. A goal-conditioned critic is referred to as a GVF [16] or UVFA [14].

Hindsight Experience Replay An untrained agent acting in a sparse reward environment rarely achieves success, which makes standard training of goal-conditioned actors and critics difficult. Hindsight Experience Replay (HER) [1] accelerates learning by augmenting real experiences in the agent's replay buffer with fake "potential" goals. The intuition behind HER is that failures are informative: a failed attempt to reach g may have led to some other potential goal g'. By pretending that g' was the agent's true goal, the agent can learn something useful even from failed attempts.

When applying HER, one must choose an appropriate sampling strategy for potential goals. The previous best strategy was the `future` strategy, which chooses potential goals to correspond to future states visited along the same trajectory. Since `future` requires us to map visited states to goals that would have been achieved in those states, we must assume that "given a state s we can easily find a goal g which is satisfied in this state" [1]. Below, we propose a novel goal sampling strategy `futureactual`, which results in state of the art performance when used together with Protoge.

2 Protoge

Desired goals in `Fetch` are completely specified by the object's (box or puck) target location—the position of the gripper is irrelevant, as are other state variables. Such goals are coarse, in that each goal corresponds to a large number of states. Using a coarse goal specification during training has at least two downsides. First, as coarser goals have larger success state sets, "achieving" a goal provides relatively less control signal. An untrained agent acting in `Push`, for example, often fails to move the box at all. In this case, the `future` strategy samples only a single potential goal, which is satisfied by *all* states in the trajectory, and little is learned. Second, it is difficult for an agent to transfer its learning to another goal space: there is no natural way for a trained `Push` agent to transfer its knowledge in order to both (1) push the box to location A, and then (2) move the gripper to location B.

To address these difficulties, we propose to train the agent on a finer—more specific—goal space and use Prototype Goal Encodings (ProtoGE) to encode coarse goals as fine ones. Formally, we say that goal space A is **coarser** than goal space B if and only if there exists **Protoge map** $f : A \to B$ such that for each goal $a \in A$, the success state set $\mathcal{S}_B(f(a))$ of its

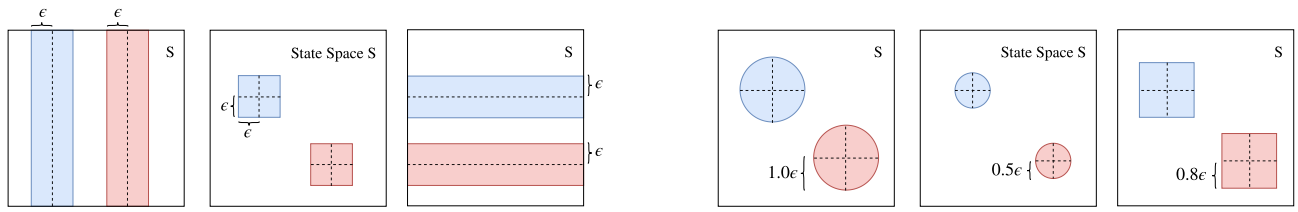

Fig. 2: Concept diagrams of *feature Protoges* (left) and *epsilon Protoges* (right). Each frame represents a different goal space over the same underlying state space, and the colored shapes inside represent success state sets for two example goals. On both the left and right, the middle goal space is finer than the adjacent spaces, which are incomparable. The goals (squares, small circles) in the middle spaces are valid Protoges for the corresponding goals (slices, large shapes) in the outer spaces.

Protoge, $f(a)$, is a subset of a's success state set $\mathcal{S}_A(a)$. If A is coarser than B, B is **finer** than A. If neither is coarser, A and B are **incomparable**. Below, we consider two types of Protoges: **feature Protoges**, which expand the dimensionality of the goal feature representation, and **epsilon Protoges**, which use tighter epsilon balls for determining goal achievement. See Figure 2 for conceptual diagrams of each. Note that any topological basis of S is finer than any goal space whose elements are open sets in S (an "open goal space"). In particular, if $S \subset \mathbb{R}^n$, the standard basis of epsilon balls about each state $s \in S$, $\mathcal{B}_\epsilon(s) = \{x \mid \|x - s\| < \epsilon\}$ where $\epsilon \in \mathbb{R}^+$, is finer than all open goal spaces. We call this the (ϵ-) **state-goal space**. Finally, note that two goal spaces A and B can both be finer than the other (as are, e.g., any two topological bases).

For example, we replace the 3-dimensional goals in `Push` with 6-dimensional goals indicating not only a target box position, but also a gripper position. Because any gripper position satisfies the original desired goal, we use a feature Protoge and require that the agent place the gripper *above* the target box location. An agent must achieve the original goal in order to achieve the Protoge. Using Protoges has at least two advantages, described below.

3 Accelerated Learning with Protoge

Although finer goals are, by definition, harder to accomplish than coarse ones, transitions between fine goals are less sparse and provide the agent with more control signal. When the gripper position is added to the agent's goal representation in the `Push` task, every movement achieves a new potential goal, even when the box remains unmoved, allowing the agent to learn how to control the gripper when using HER's `future` strategy. The intuition here is similar to that of auxiliary tasks [6] and GVFs [16]: learning to control things—even if not directly connected to the agent's primary goal—results in better overall control and improved performance. Our results suggest that a balance between the additional difficulty and control information introduced by specificity can accelerate learning.

Fetch Results We demonstrate accelerated learning in `Push`, `PickAndPlace` and `Slide` by (1) expanding the dimensionality of the `Push` and `PickAndPlace` goal spaces using feature Protoges, and (2) increasing the specificity of the `Slide` goal space using an epsilon Protoge. Test success is always measured with respect to the original goal space. See

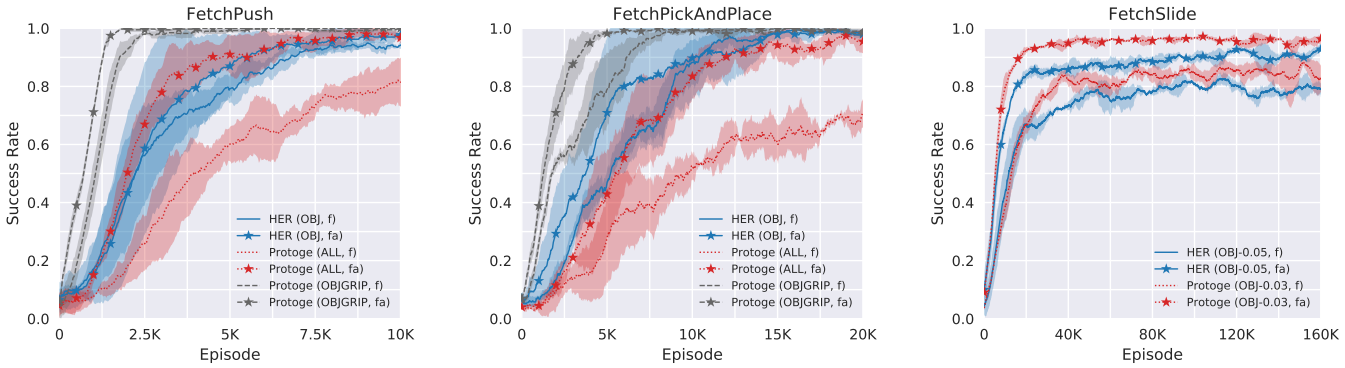

Fig. 3: In `Push` and `PickAndPlace`, the standard goal representation `OBJ` is greatly outperformed by the finer `OBJGRIP` representation, which uses 6-dimensional Protoges. The `ALL` state-goal representation does not do as well as `OBJGRIP`, but is able to a learn even when thresholds are hard-coded. In `Slide`, our epsilon Protoge (`OBJ-0.03`) approach outperforms the standard `OBJ-0.05` approach. In all cases, the `futureactual` strategy (fa) performed better than the `future` strategy (f). When used together, Protoge and `futureactual` establish a new state-of-the-art. Our results can be compared to the baselines of Plappert et al. (2018) [13] by noting that our 10K episodes are roughly equal to 5 Plappert epochs. Our greatly improved baseline results are due to implementation differences (see main text). For all environments, each configuration was run with the same 3 random seeds.

Figure 3. In `Push` and `PickAndPlace` we experiment with a 6-dimensional object-gripper goal specification (`OBJGRIP`), which reports both the box position and the gripper position, and a 25-dimensional state-goal specification (`ALL`), which reports all 25-dimensions of the state. In each case we use a hand-coded Protoge map to encode desired goals into the expanded goal space. The Protoge places the hand above the box, and encodes the other dimensions (e.g., speed and object rotation) in the state-goal case to have a value of 0. We use the same epsilon threshold as the original environments with respect to the original goal dimensions (thus, the expanded goal space is finer than the original goal space, and achieving an expanded goal necessarily achieves the original goal), and use an element-wise epsilon threshold for added dimensions, which we manually set to 0.05. In `Slide` we experiment with a tighter goal specification, which reduces the distance threshold for goal satisfaction from 0.05 to 0.03 (meters). We plan to experiment with automatically learning optimal thresholds in future work. We expect that learned thresholds will greatly improve performance, especially that of the state-goal representation (`ALL`).

A Novel HER Strategy As a result of the increase in goal specificity, the agent's state rarely satisfies the Protoge of any potential desired goal (e.g., the agent's gripper is rarely directly above the box); as such, when using the `future` strategy, very few desired goal Protoges are added to the agent's replay buffer. To combat this effect, we propose the `futureactual` strategy, which mixes goals sampled according to `future` with goals sampled from a buffer of past *actual* goals (in the agent's goal space; i.e., using Protoge). This focuses the agent's learning effort on actual desired goals while still providing the agent with enough initial reward signal to benefit from HER. In our experiments we always sample 80% of the agent's training experiments using HER, with 40% sampled according to `future` and 40% sampled according to `actual`. Although Protoge still works with `future`, we find that `futureactual` improves performance, even in absence of Protoge (Figure 3).

Implementation Highlights We use DDPG together with our own implementation of HER. Rather than distribute training across parallel workers (as done by Plappert et al. [13]), we train a single agent in 12 parallel environments. Our actors and critics use 3 layer-normalized [2] hidden layers of 512 units each. We train with a batch size of 1000 every two environment steps, and update our target network every 40 training steps using an update factor of 0.05. We apply L_2 action normalization with coefficient 0.1. Our `Push` and `PickAndPlace` agents use epsilon exploration with an epsilon of 0.3, whereas our `Slide` agent does not use any epsilon exploration. Other hyperparameters are similar to those used by Plappert et al. [13]. Our baseline `future` agents learn significantly faster than the agents of Plappert et al., as well as the more recent agents of Liu et al [8], most likely due to the different training regime.

4 Transfer Learning with Protoge

Finer goal representations are more flexible and allow for efficient transfer. The state-goal representation in particular, is *universal*: an agent trained on the state-goal space can potentially adapt to arbitrary goals, regardless of their form, so long as Protoges are available. This effectively reverses the HER assumption: instead of assuming that given state s we can find a goal g that is satisfied in s [1], we assume that given goal g we can find a state s that satisfies g. Our present work demonstrates transferability using hand-designed Protoge maps, but we are working towards an effective method for learning Protoges maps online.

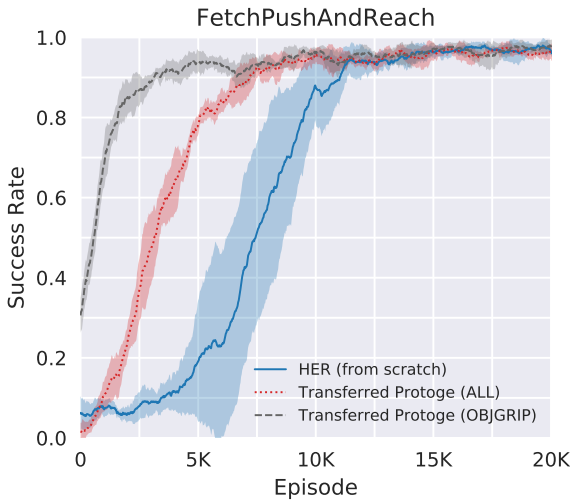

Fig. 4: Transferred Protoge agents, trained on `Push`, solve `PushAndReach` faster than an agent trained from scratch (5 random seeds each).

In Figure 4, we show that both the `OBJGRIP` and `ALL` agents from the previous Section can effectively transfer their knowledge to a `PushAndReach` environment, which has 6-dimensional goals and requires the agent to both (1) push the box to a 3-dimensional target location and (2) move its gripper to another 3-dimensional target location. The target locations are independently sampled (the box goal is sampled according to the same distribution as `Push`, and the gripper goal is sampled roughly according to the same distribution as the `FetchReach` environment). Since `PushAndReach` is harder than `Push`, we see that the standard HER approach learns slightly slower than it does in `Push`. The transferred agents are able to learn much faster, even though the distribution of desired goals has changed significantly. Note that while the transferred `OBJGRIP` agent is now using the native goal space (no Protoge), the transferred `ALL` agent is using a Protoge map, as before, to expand the native 6-dimensional goal into a 25-dimensional goal.

5 Future Work

This work is still in its early stages, and we are exploring several avenues going forward:

- In our `Fetch` experiments, we use a hard-coded element-wise success threshold for expanded goal dimensions. Can we learn these thresholds automatically? We are currently exploring a curriculum learning approach [5].
- In particular, many aspects of the state space are usually outside of the agent's control, and it is unreasonable to ask an agent to achieve arbitrary goals in the state-goal space. How can we automatically recognize controllable aspects of the state and use only those when defining Protoges? One promising approach is to learn a "controllable" latent space by predicting inverse dynamics $f(a_t|s_t, s_{t+1})$ [12].
- In our `Fetch` experiments, we use the native feature space to define goals and Protoges. How can we generalize this approach to the agent's latent feature space?
- More generally, we may wish to learn Protoge maps between two complex goal spaces (recall that two goal spaces can both be finer than the other). For example, we might want to map the natural language goal space [4], describing the task that the agent needs to perform, to raw pixels (an image) showing the agent's first person view [11], or vice versa. How can we learn such maps automatically?
- For a given coarse goal, there are many candidate Protoges, any of which satisfy the original goal. It would be interesting explore the generation of optimal Protoges, conditioned on the current state and goal.
- In our experiments, we used the original `Push` goal space to generate Protoges in order to train the `ALL` agent, which agent was able to quickly transfer its knowledge to `PushAndReach`. It seems likely, however, that the `ALL` agent could have designed its own goal curriculum, separately from `Push`, and trained itself in an unsupervised fashion by taking advantage of, e.g., curiousity [12].

References

[1] ANDRYCHOWICZ, M., WOLSKI, F., RAY, A., SCHNEIDER, J., FONG, R., WELINDER, P., MCGREW, B., TOBIN, J., ABBEEL, O. P., AND ZAREMBA, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems* (2017), pp. 5048–5058.

[2] BA, J. L., KIROS, J. R., AND HINTON, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[3] BROCKMAN, G., CHEUNG, V., PETTERSSON, L., SCHNEIDER, J., SCHULMAN, J., TANG, J., AND ZAREMBA, W. Openai gym, 2016.

[4] CHAN, H., WU, Y., KIROS, J., FIDLER, S., AND BA, J. Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning. *arXiv preprint arXiv:1902.04546* (2019).

[5] EPPE, M., MAGG, S., AND WERMTER, S. Curriculum goal masking for continuous deep reinforcement learning. *arXiv preprint arXiv:1809.06146* (2018).

[6] JADERBERG, M., MNIH, V., CZARNECKI, W. M., SCHAUL, T., LEIBO, J. Z., SILVER, D., AND KAVUKCUOGLU, K. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397* (2016).

[7] 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. *arXiv preprint arXiv:1509.02971* (2015).

[8] LIU, H., TROTT, A., SOCHER, R., AND XIONG, C. Competitive experience replay. In *International Conference on Learning Representations* (2019).

[9] LOCKE, E. A., SHAW, K. N., SAARI, L. M., AND LATHAM, G. P. Goal setting and task performance: 1969–1980. *Psychological bulletin 90*, 1 (1981), 125.

[10] MNIH, V., KAVUKCUOGLU, K., SILVER, D., GRAVES, A., ANTONOGLOU, I., WIERSTRA, D., AND RIEDMILLER, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[11] NAIR, A. V., PONG, V., DALAL, M., BAHL, S., LIN, S., AND LEVINE, S. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems* (2018), pp. 9209–9220.

[12] PATHAK, D., AGRAWAL, P., EFROS, A. A., AND DARRELL, T. Curiosity-driven exploration by self-supervised prediction. In *ICML* (2017).

[13] PLAPPERT, M., ANDRYCHOWICZ, M., RAY, A., MCGREW, B., BAKER, B., POWELL, G., SCHNEIDER, J., TOBIN, J., CHOCIEJ, M., WELINDER, P., ET AL. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464* (2018).

[14] SCHAUL, T., HORGAN, D., GREGOR, K., AND SILVER, D. Universal value function approximators. In *International Conference on Machine Learning* (2015), pp. 1312–1320.

[15] SUTTON, R. S., AND BARTO, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[16] SUTTON, R. S., MODAYIL, J., DELP, M., DEGRIS, T., PILARSKI, P. M., WHITE, A., AND PRECUP, D. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2* (2011), pp. 761–768.