Speeding-up Continual Learning through Information Gains in Novel Experiences

Pierrick Lorang[*1,2], **Shivam Goel**[*3], **Patrik Zips** [2], **Jivko Sinapov**[3], **Matthias Scheutz**[3,2]

[1]Department of Mechanical Engineering, Tufts University, Medford, USA
[2]AIT Austrian Institute of Technology GmbH, Center for Vision, Automation & Control, Vienna, Austria
[3]Department of Computer Science, Tufts University, Medford, USA
{pierrick.lorang, shivam.goel, jivko.sinapov, matthias.scheutz}@tufts.edu; patrik.zips@ait.ac.at

Abstract

Adapting to novelties in open-world environments is an important and difficult challenge, and it has been recently shown that hybrid planning and reinforcement learning approaches can lead to better adaptations. However, these approaches still face intriguing difficulties induced by their heavy dependence on training samples to overcome changes in the environment quickly. In this work, we propose an integrated planning and learning approach that utilizes learning from failures and transferring knowledge over time to overcome novelty scenarios. Our proposed approach is much more sample efficient in adapting to sudden and unknown changes (i.e., novelties) than the existing hybrid approaches. We showcase our results on a Minecraft-inspired gridworld environment called *NovelGridworlds* by injecting *three* novelties in the agent's environment at test time. We show that our approach can speed up continual learning through information gained in each novel experience and, thus, more sample-efficient.

Introduction

One of the most pressing unresolved challenges to AI learning algorithms like reinforcement learning (RL) is how to cope with novelties (e.g., Muhammad et al. (2021)). Novelties differ from new data points outside a given distribution or anomalies along expected dimensions in that they usually do not fit into an agent's representational space and often require the agent to completely change its problem representations to accommodate them. In particular, the agent's initial knowledge base cannot solve the problem once the environment changes (Sarathy and Scheutz 2018). Hence, the questions arise (1) how to detect and characterize novelty, and (2) when and how to accommodate novelties in ways that are beneficial to the agent as to improve its future performance. Answering these questions is critical for agents operating in "open worlds" where they are almost guaranteed to face novelties during their task performance, some of which might have detrimental effects on the agent's performance if the agent does not know how to deal with them, while some might present opportunities for performance improvements if the agent can figure out how to utilize them. Most importantly, while the standard response to novelties in domains is to collect more data or to train from scratch (as is common with RL algorithms), it is often not possible for agents performing tasks to extensively explore potential changes to their representation of the environment. Hence, our goal is not only to enable agent to handle novelties, but also to handle them *as quickly as possible*, ideally with the minimum amount of "regret" to the learner.

In this paper, we will explore how our previously proposed algorithm for rapidly accommodating novelties in an RL settings -RAPid-Learn (Goel et al. 2022)- can be extended to perform better over encounters with multiple novelties and thus make progress towards fast continual learning. We begin by motivating our approach and reviewing related work, then present a simple working example to introduce the concept. We follow by explaining some preliminaries and necessary background before turning to the problem formulation and our proposed solution. To conclude, we describe our experience, baseline comparisons and results which we finally discuss as well as provide our future vision and improvements on this approach.

Motivation: Fast Life-Long Learning[*]

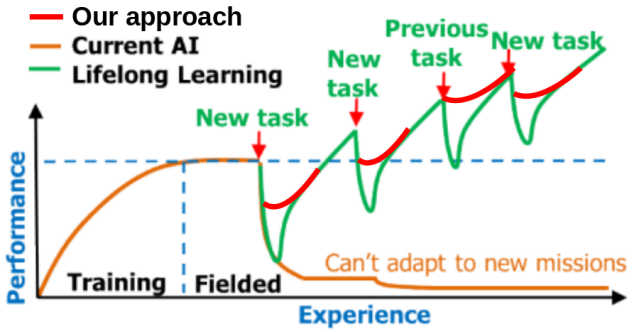


Figure 1: The impact of new task on performance is significantly reduced in the agents proposed in this paper compared to typical life-long learning agents and there are no negative effects when previously learned tasks need to be performed.

[*]These authors contributed equally to this work

[*]This image is adapted from the BAA for the DARPA SHELL program

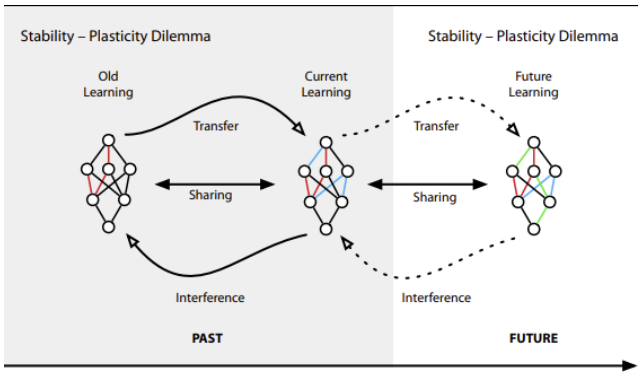


Figure 2: "The stability-plasticity dilemma considers plasticity with respect to the current learning and how it degrades old learning. The transfer-interference trade-off considers the stability-plasticity dilemma and its dependence on weight sharing in both forward and backward directions." (Riemer et al. 2019).

Related Works

Current approaches to continual ("life-long") learning are typically variants of *deep reinforcement learning* (Khetarpal, Riemer, and amd Doina Precup 2020). Studied as a possible solution on their own for continuous (Abel et al. 2018; Lecarpentier et al. 2020) and non-stationary (Cheung, Simchi-Levi, and Zhu 2020) learning, they have shown catastrophic forgetting problems and difficulty in dealing with abrupt novel events, requiring to learn again from scratch while ideally they should be able to build on their previously learned knowledge as we propose (see Fig. 1).

The main challenge for these approaches is a tension between forgetting previously learned tasks when learning news tasks and being able to learn new tasks quickly and also generalize to future tasks (see Fig. 2). Specifically, the common approach of weight sharing in neural architectures across different learning agents, especially when their tasks are very different, does not benefit continual learning because it exacerbates this tension and requires agents to have similar Deep Neural Network (DNN) architectures (i.e., knowledge cannot be shared with other non-DNN agents).

Various methods have been proposed to address this tension (e.g., Meta Experience Replay (MER) (Riemer et al. 2019) or "powerplay" (Schmidhuber 2013)), but they all require that previously used task traces for learning skills be retained for training to prevent forgetting which is not realistic for the limited computing and storage capability of edge computing systems.

One way to help address the problem is to use symbolic and semantic knowledge to control and store the information acquired during each reinforcement learning cycle. Symbolic planning is an effective way to solve problems when efficient plan operators are available (Ghallab, Nau, and Traverso 2016). However, since plan operators are typically defined by hand, aside from being laborious, it is impossible to anticipate future changes to the environment. This last problem is all the more important to treat in environments where actions represented by operators are prone to unexpected failures or malfunction due to novelties.

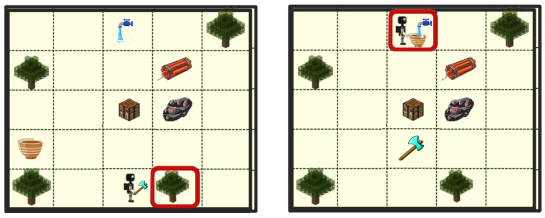
We have shown that a hybrid approach using both reinforcement learning and symbolic planning techniques, can take advantage of both methods (Sarathy et al. 2021). In this approach we used a tabular RL to solve plan failures induced due to novelties. In our more recent work on integrating planning and learning, we further improved the novelty accommodation at plan execution failures at run-time (Goel et al. 2022). We instantiate a neural network to learn action executors of the failed operators in the form of RL policies on-the-fly. This method also performs knowledge-guided-exploration to learn policies rapidly.

Yet, using RL to recover from planning failures remains a challenge notably because RL methods require a large number of training samples. The Rubik's cube solving robot developed by OpenAI, Dactyl (Akkaya et al. 2019), ran an RL algorithm in simulation for the equivalent of 10,000 years in real time during training, resulting in a success rate of 20% for a maximally scrambled cube. Handling novelties requires responsiveness using very little data. There is a clear need to improve the efficiency of learning and to use every piece of information acquired through experience. In this paper, we propose to integrate the Hindsight Experience Replay (HER) technique (Andrychowicz et al. 2017) into our hybrid planning and learning approach. HER has demonstrated sample and reward efficiency over traditional RL techniques. It showed that it is possible to leverage failures during task training using a goal-oriented policy and a replay buffer. HER has strong potential to reuse information for sample efficiency and thus accelerate learning over time. We evaluate our methodology in the Minecraft-gridworld inspired domain (Goel et al. 2021) by injecting several novelties in the agent's environment during test times.

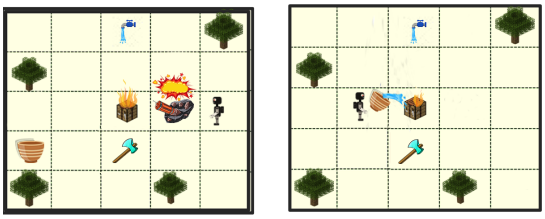
Running Example

Achieving a goal that is not the desired goal still gives us information about how to achieve a state in the environment that could be used in the future (Andrychowicz et al. 2017). Let us take a task that an agent has to accomplish in a Minecraft-like environment: *craft a pogostick*. The task involves the agent collecting items (*tree logs, iron*) in the environment which can be crafted into other things like *planks, rubber, pogostick*. However, there may be unexpected events (novelties) that challenge the success of each operator that play a role in the agent's plan.

The example represented in Figure 3a depicts a novelty scenario where the $trees$ in the environment require *axe* to cut. While exploring the environment, the agent may try actions to find a solution to retrieve *tree logs* from *trees*. It will focus on the desired goal of fetching a *tree log*, which will require several samples for training, and the agent will fail in most of them. Yet, even if it fails in achieving the desired goal, the agent reaches other states, called "achieved goals", for example, in some of them it might use a concave object in the environment to store more water than it usually does. While this information is not useful for the current task, it can be crucial for learning how to put out a fire if it appears as a novelty later on (shown in Figure 3b).



(a) Left: Desired goal: get a *tree log* using the *axe*. Right: Achieved Goal: the agent filed a concave object with more *water* than it usually can store.



(b) Left: Blowing the *iron ore* put the *crafting table* on *fire*. The agent needs to put it out. Right: The agent utilize crucial information from previously failed trials to solve this novelty faster.

Figure 3: Illustration of the Running Example

Preliminaries

Reinforcement Learning

Reinforcement learning (RL) (Sutton, Precup, and Singh 1999) is a field of machine learning in which an intelligent agent (computer program) interacts with the environment and learns to act in that environment based on a reward system. An RL problem is generally formulated as a Markov Decision process (MDP). An MDP $M = \langle S, A, R, T, \gamma \rangle$ consists of finite set of states S and actions$A$, a transition function $T : S \times A \rightarrow S$ defining a set of transition probabilities $p(s'|s, a)$, and a reward function $R : S \times A \rightarrow \mathbb{R}$ mapping state-action pairs to scalar rewards. The goal of such algorithms is to maximize the expected sum of future discounted rewards, with the importance of future rewards weighted by a discount factor $\gamma \in [0, 1)$. This is known as expected discounted return which, at time t, is: $Gt = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$.

The value of each state-action pair under a policy π is defined as the expected return when starting in s and following π afterwards, $Q^{\pi}(s, a) = E_{\pi}[G_t|S_t = s; A_t = a]$. An optimal policy is defined as any policy π^* so that $Q^{\pi^*}(s, a) \geq Q^*(s, a)$ for every possible state, action, and policy. This optimal Q-function Q^{π^*} can be shown to satisfy what is known as the Bellman equation: $Q^{\pi^*}(s, a) = E_{s' \ p(.|s,a)}[R(s, a) + \gamma max_{a' \in A} Q^*(s', a')]$.

Universal Value Function Approximators (UVFA)

Classically in RL, a value function is geared towards a single goal and encodes a chunk of information about the environment that is utilized to reach it. UVFA idea (Schaul et al. 2015) is to represent a large set of these value functions using just one unified function approximator generalizing over both states and goals. Just as general value functions approximators are able to exploit state space structure to accurately generalize to similar previously unseen states, the same is true across goals (Schaul et al. 2015). The insight behind this technique is that goals are specified using the same predicates that describe the states. UVFAs have been shown to exploit both similarity encoded in the formulation of goal representations themselves as well as structure of the specific value functions discovered for each goal.

With UVFA, every episode has a static goal for the entirety of the episode. At every time-step within that episode, the agent receives not only the current state as input but also the current goal $\pi : S \times G \rightarrow A$. The resulting reward is now also goal-dependent $r_t = r_g(s_t, a_t)$ and the corresponding Q-function now involves a goal in addition to the state-action pair $Q^{\pi}(s_t, a_t, g) = E[R_t|s_t, a_t, g]$. Schaul et al. show that it is possible to train an approximator to the Q-function with direct bootstrapping using a variant of Q-learning: $Q(s_t, a_t, g) \leftarrow r_g + \gamma_g max_{a'} Q(s_{t+1}, a', g) + (1 - \alpha)Q(s_t, a_t, g)$. A direct consequence of this result is that UVFA can be used alongside algorithms like DQN and DDPG. It has been shown through experiments in the Lava-World (Leike et al. 2017) environment that UVFA is able to generalize well across state-goal pairs given a sufficient amount of experience.

Hindsight Experience Replay (HER)

Hindsight Experience Replay (HER) uses an off-policy algorithm, like Deep Q-Network (DQN) (Mnih et al. 2013) and combines it with the multi-goal RL formulation introduced by UVFA. It also modifies the sampling procedure. The multi-goal formulation means that rewards are now goal-specific and input to the policy includes the current goal concatenated onto the corresponding state observation at each time step. As explained previously, HER performs experience replay, but after every episode $(s_0, s_1, ..., s_T)$ when it stores the corresponding transitions $(s_t \rightarrow s_{t+1})$ experienced within the episode, it stores each transition with a subset of other goals in addition to the original one.

HER assumes that every goal $g \in G$ corresponds to some predicate $f_g : S \rightarrow \{0, 1\}$, assumption that we follow in this paper, and that for a given goal the agent attempts to reach any state where $f_g(s) = 1$. It also assumes that we have a mapping from states to goals identifying a goal that is satisfied in that state (in the case where a goal corresponds to a state we want to achieve, this mapping is an identity). Replay trajectories can be simulated with an arbitrary goal using an off-policy RL algorithm. Through the imagination of trajectories with different goals, HER aims to learn multiple variations of the same task from a single experience. Experimental results show that HER improves performance even with a single goal, though training on multiple goals results in faster training even in the single goal case.

Integrated Planning and Learning for Open-World Novelty Handling

We extend this work from our previous methodologies (Goel et al. 2022; Sarathy et al. 2021) which combines symbolic

planning and targeted reinforcement learning to deal with plan execution failures due to novelties in the agent's environment. The agent is assumed to start with a domain knowledge, grounded using PDDL (McDermott et al. 1998), and defined as $\sigma = \langle \mathcal{E}, \mathcal{F}, \mathcal{S}, \mathcal{O} \rangle$, where $\mathcal{E}$ is a finite set of known entities within the environment. $\mathcal{F}$ is a finite set of known predicates with their negations. $\mathcal{S}$ is the set of symbolic states in the environment. $\mathcal{O}$ denotes the set of known action operators.

In the more recent work (Goel et al. 2022, 2021) we define novelty as a new encounter that breaks the known dynamics of the environment or doesn't belong to the agent's domain knowledge and that can not be inferred by the agent cognitive abilities. A novelty is detected via an operator in the plan that, at execution time, can't be executed or doesn't produce the expected effects and fails; we call it a failed operator.

Definition 1 *(Novelty). Considering $\mathcal{E}'$: the set of novel entities in the environment such that $\mathcal{E}' \cap \mathcal{E} = \emptyset$, $\mathcal{S}'$: the set of novel states such that $\mathcal{S}' \cap \mathcal{S} = \emptyset$, $\mathcal{O}'$: the set of novel operators such that $\mathcal{O}' \cap \mathcal{O} = \emptyset$, and $\mathcal{F}'$ a set of novel predicates which are unknown to the agent, a novelty is defined as a tuple $N = \langle \mathcal{E}', \mathcal{F}', \mathcal{S}', \mathcal{O}' \rangle$.*

A novelty introduced by the environment results in inadequate domain knowledge (Section *Novelties*), causing an execution impasse due to operator failure. To overpass this impasse we define a stretch-Integrated Planning Task (Stretch-IPT) to recover these missing operators from the agent's knowledge base.

Definition 2 *(Integrated Planning Task) An Integrated Planning Task (IPT) is $\mathcal{T} = \langle T, \mathcal{M}, d, e \rangle$ where $T = \langle \mathcal{E}, \mathcal{F}, \mathcal{O}, s_0, s_g \rangle$ is a STRIPS task, $\mathcal{M}$ is the set of MDPs. A detector function $d : \widetilde{\mathcal{S}}' \rightarrow \mathcal{S}$, where $\widetilde{\mathcal{S}}'$ is the set of sub-symbolic states, determines a symbolic state for a given sub-symbolic MDP state, and an executor function $e : \mathcal{O} \rightarrow \mathcal{X}$, $\mathcal{X}$ being the set of executors, maps an operator to an executor (Sarathy et al. 2021).*

Definition 3 *(Stretch-IPT). A Stretch-IPT $\tilde{\mathcal{T}}$ is an IPT $\mathcal{T}$ for which a solution exists, but a planning solution does not.*

We are interested in finding a solution to the stretch-IPT, specifically study how to automatically generate executors on-the-fly to solve the execution impasse.

Definition 4 *(Executor Discovery Problem). Given a stretch-IPT $\tilde{\mathcal{T}} = \langle T', \mathcal{M}', d', e' \rangle$ with $T' = \langle \mathcal{E} \cup \mathcal{E}', \mathcal{F} \cup \mathcal{F}', \mathcal{O} \cup \mathcal{O}', s'_o, s_g \rangle$, construct a set of executors $\{x'_1, \ldots, x'_m\} \in \mathcal{X}$ for the set of failed operators $\{o_1, \ldots, o_m\} \in \mathcal{O}$ such that the stretch-IPT $\tilde{\mathcal{T}}$ is solvable, with the executor function:*

$$e'(o_i) = x'_i, o_i \notin \mathcal{O} \tag{1}$$

i.e., we find an executor whose operator does not exist in $\mathcal{O}$.

According to our previous work, we define each action executor which consists of a tuple $\langle I, \pi, \beta \rangle$. It is similar to the options framework as defined by Sutton, Precup, and Singh (1999). We formally define the action executors as

Definition 5 Each action executor x_i is represented by a tuple $<I, \pi, \beta>$, where $I \subseteq S$ is the initiation set, denoting the set of states when the action executor X_i is available for execution, and it follows a policy $\pi : S \times A \rightarrow [0, 1]$ to get to the termination state $\beta \subseteq S$.

Proposed Framework

Goal-conditioned Executor

Our proposed solution is a hybrid method of planning and RL using a goal-oriented policy and transferring policy networks between novelties. We leverage both classical RL transfer learning and information gained from failures using HER technique with DQN. We perform RL transfer learning by transferring policy networks, i.e., we copy the weights from the neural architecture of the source policy network and initialize the newly considered policy network with these values. The difference in DQN transfer learning process compared to HER is that HER policy networks also consider the goal as a neural input in addition to the state.

We begin by augmenting the definition of action executor x_i (Definition 5) to include goals in the policy π. We formally define the goal-conditioned action executor as:

Definition 6 *(Goal-conditioned Executor)* Each goal-conditioned action executor x'_i is represented by a tuple $<I, \pi'_{s_{g_i}}, \beta>$, where $I \subseteq S$ is the initiation set, denoting the set of states when the action executor X'_i is available for execution, and it follows a goal-conditioned policy $\pi'_{s_{g_i}} : S \times G \rightarrow A$ which terminates when $f_g(s) = 1$, where goal $g \in G$ corresponds to some predicate $f_g(s) : S \rightarrow \{0, 1\}$. *The policy π' terminates when the goal g_i is satisfied. $\beta \subseteq S$ is the set of termination states of the executor which satisfies the condition $f_g(s) = 1$.*

For example, if the agent's goal is to fetch *axe* in the inventory, the executor can be conditioned on the goal of getting an axe in the inventory, which can later generalize to a new goal of fetching other items (such as *water*) in the inventory. This gives us flexibility of evolving the policy over time on new goals and generalizing the executor over novelties.

Executor Evolution

The goal-conditioned executor is evolving over time by generalizing to new goals and transferring the old policy as the initial policy for the new goal. Formally, for a new goal g_i we initialize an executor x'_i whose starting state I is the current operator's failed state and whose termination set of states β is the set of states that satisfies the predicate $f_{g_i} = 1$. The policy $\pi'_{s_{g_i}}$ of x'_i, is initialized using the trained policy on previous experiences $\widetilde{\pi}'_{s_{g_i}}$.

As described above, we continually evolve the executor over time as the agent experiences execution impasses induced by novelties in its environment. Based on new goals, we transfer the previous policy $\widetilde{\pi}'_{s_{g_i}}$ as a initial policy for the new goal $\pi'_{s_{g_i}}$ and perform continual learning.

We use the executors (as defined in Def. 6) and continually evolve them by transfer learning. The aim is to gain efficiency by reusing information acquired during experimenta-

tion and to continually improve the agent's reaction to novelties. Hence, we aim at continually improving the learning metrics, i.e., higher jump-start, shorter time-to-threshold, and better asymptotic performance, while providing a solution to the source task selection problem in transfer learning using a goal-dependent policy network. In this work, however, we do not explicitly solve the problem of source task selection while transferring the policy networks and therefore could be an interesting direction of future work.

Dealing with Continual Domain Growth

To handle continual domain growth when using a fixed neural network structure like a DQN, we put placeholders in the action and state spaces. We also anticipate goal placeholders as HER needs an explicit definition of the goal to compare achieved goals with desired goals.

In continual learning, HER is not used the traditional way in which the gains of a multi-goal problem formulation are immediately observable. In contrast each novelty goal formulation is treated separately and continually, as would be the case in an open world, and we can not anticipate an explicit definition of the novelties goals. To respond to this problem we propose a relaxed version of the reward function given in the original HER paper, that we define at each time step t as:

$$R_{g_i}(s_t, a_t) = \begin{cases} 0, & \text{if } s_t \subseteq s_{g_i} \\ -1, & \text{otherwise.} \end{cases} \quad (2)$$

Experiments

We use the NovelGridworlds (Goel et al. 2021) environment to conduct our experiments. The task we focus on is called *crafting a pogostick*. The agent starts with complete knowledge of the pre-novelty domain grounded in the form of PDDL (McDermott et al. 1998). The agent uses the metricFF (Hoffmann 2003) planner to generate a plan to solve the task. The environment is injected with novelties which causes the operators in the plan to fail at execution time. The agent then formulates an MDP on-the-fly and solves this MDP resulting in learning an action executor (as defined in Definition 5). Once the agent learns to succeed in the impasse caused by the novelty, it can switch back to the original plan to execute the operators and complete the task. In our experiments, we design novelties that can cause further impediments for the agent to solve the task. We utilize our approach in transferring the policy networks to learn new policies for future operator failures in the plan execution. We describe the task environment in detail, followed by the novelties injected. We then describe the evaluation metrics and the baselines to compare our method.

NovelGridworlds

NovelGridworlds is an openAI gym (Brockman et al. 2016) gridworld environment inspired from Minecraft where the agent can collect items in its inventory, and craft subsequent objects using various crafting actions (A diagrammatic representation is shown in Figure 4). The observation space consists of information from a LiDAR-like sensor that emits

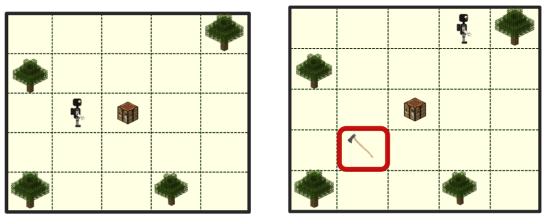


Figure 4: Left: Pre-novelty domain showing an agent which can craft items by breaking *trees*. Right: Novelty induces an impasse which requires the agent to use the axe (shown in red) to break the *trees*.

beams for each entity (e.g., tree, crafting-table, wall etc.) in the environment at incremental angles of $\frac{\pi}{4}$ to determine the closest entity in the beam angle, in other words, the LiDAR-like sensor provides an observation of size $8 \times |\epsilon|$, where ϵ is the set of possible entities. Additional sensors observe the contents of the agent's inventory and the currently selected entity. The action space is the sub-symbolic action space given by the environment (navigation actions: turn left, turn right, move forward; interaction actions: break, extract rubber; craft actions: craft planks, craft sticks, craft pogo stick), augmented with novel actions and hierarchical action operators (entity-parameterized approach-$\langle$entity$\rangle$) implemented by the planner. The domain was chosen for two main reasons:

- It provides a complex task that involves a sequential set of actions to reach the final state. If the agent misuses its resources, it will fail to accomplish the task;
- The domain is designed specifically for solving for solving problems in an open-world, which allows us to create and experiment with a variety of novelties (Goel et al. 2021). Items available in the RL environment must be differentiated from items acknowledged by the agent at plan-time with which it can already plan.

Novelties

The novelties in the experiments are induced by changes in the dynamics of the environment which result in plan execution failures. We implement three novelty scenarios.

Axe-to-break In the pre-novelty scenario, the agent can break *trees* in the environment by using *hand*. In the novelty scenario, the *trees* become harder to cut, and the agent needs to carry and select an *axe* next to a *tree* to get *tree-logs* (illustrated in Figure 5). Therefore, at plan execution time, the operator *break tree-log* fails and the agent needs to explore the environment to learn the new dynamics of fetching an *axe* and using it to cut *trees*.

Dynamite The second novelty, illustrated in Figure 6, is that the the agent now needs *iron* as an extra ingredient to craft a *pogostick*. The initial recipe of crafting a *pogostick* does not contain *iron*. Now the agent needs to blow an *iron-ore* using a *dynamite* and fetch *iron*. It can then use *iron* to craft a *pogostick*. Therefore at the plan execution time, the operator *craft-pogostick* fails, and the agent needs to explore the environment to learn these new dynamics.

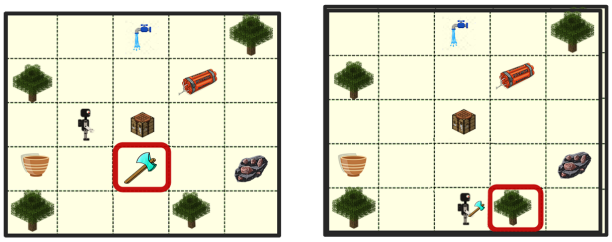


Figure 5: Illustration of *axe to break* novelty. Left: Breaking a *tree* now requires an *axe*, available in the environment. Right: If the agent carries this *axe* in its inventory and selects it in front of a *tree*, it fetches *tree logs*.

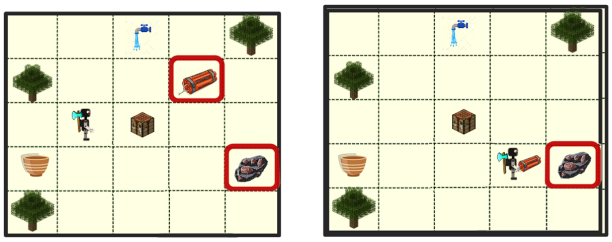


Figure 6: Illustration of *dynamite* novelty. Left: The actual recipe to craft a *pogostick* includes *iron*, which can only be obtained by blowing the *iron ore* using a *dynamite*. Right: The agent will blow the *iron ore* resulting in 4 more *iron bars* in its inventory but leading to Novelty3.

Fire In this novelty scenario (illustrated in Figure 7), the *crafting-table* is set on *fire*, due to the blowing of *dynamite*. This prohibits the agent's use to craft anything using the *crafting-table*. In order to use the *crafting-table* again, the agent needs to use *water* to put it out. The agent can only obtain one quantum of *water* from the *water tap*. To obtain *water*, the agent now needs to use a *concave object* to fill *water*, and then use it to put out *fire* from the *crafting table*. Therefore, in this novel scenario the operator *craft-pogostick* will fail, and the agent needs to explore the environment to learn the above mentioned dynamics.

Evaluation Metrics

We evaluate our results based on the success rate of the overall policy while training on each novelty on a test set averaging the results over 10 different training seeds. We also consider the number of steps to converge to a given success rate threshold of 0.99. Each 500 training steps we run 300 evaluation episodes to score the policy on a test set. The initial map in which the agent starts in each seed is random (but seed dependent).

Baselines

We call our approach as **RAPid-Learn+HER+TL** (Integrated planning and learning combined with HER and Transfer learning), and compare it against four baselines:

- **HER**: HER alone on the entire task of crafting a *pogostick* that learns to reach its goal and adapts its RL policy to the three novelties.

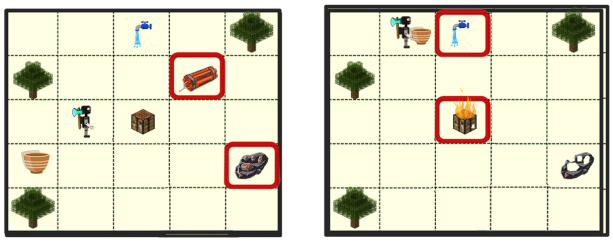


Figure 7: Left: Again the actual *pogostick* recipe includes *iron*, which can only be obtained by blowing the *iron ore* using a *dynamite*. Yet the *crafting table* catches fire from the dynamite blow. It needs 5 quantums of *water* to put it out yet out agent can only carry one quantum of *water* at a time. Right: Using the *concave object* present in the environment will help him carry the five quantums of *water*, put out the fire and finally craft the *pogostick*.

- **RAPid-Learn+DQN**: The hybrid approach of using planning and DQN as the RL algorithm to learn to overcome novelties. We don't do any policy transfer learning here.
- **RAPid-Learn+DQN+TL**: The hybrid approach of using planning and DQN as the RL algorithm to learn to overcome novelties while performing transfer learning.
- **RAPid-Learn+HER**: The hybrid approach of using planning and HER(Nair et al. 2017) as the RL algorithm to learn to overpass novelties, without any policy transfer learning.

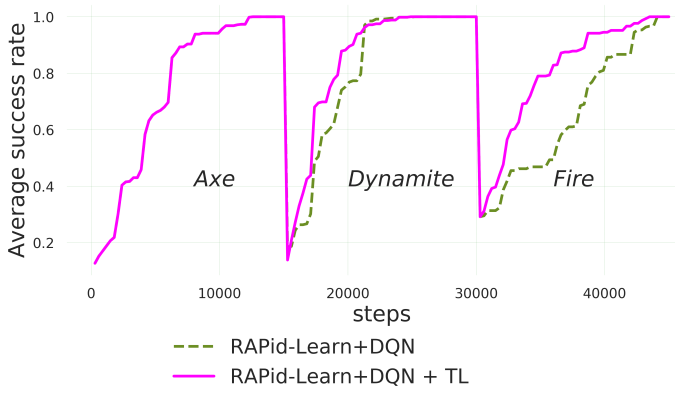
We will be able to compare the performance of hybrid planning with HER against these baselines and analyze the features from which it benefits. It will notably allow us to see how good HER can take advantage of information gained from previous failures.

Results & Discussion

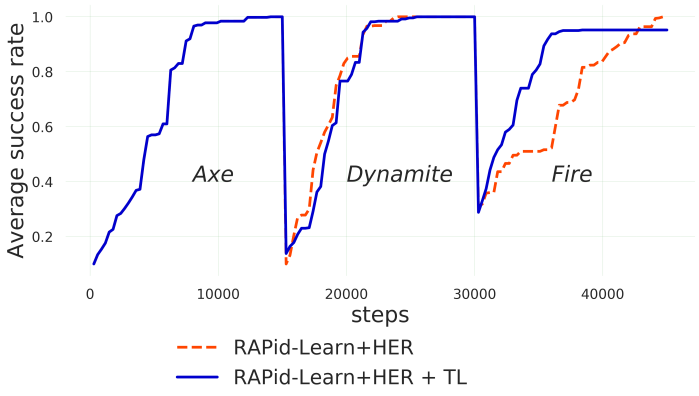
We plot the learning curves of each agent's performance on all the novelties. We show a combined learning curve (shown in Figure 8) to showcase how transfer learning and learning from failures makes novelty adaptation better over time. Figure 8b shows that our approach improves the novelty adaptation by improving the time to threshold (specially on later novelty). It can be clearly seen that our method successfully reuses information from previous trials to speed up the learning and shows an overall higher success rate during training on new novelties(Figure 10b).

The more novelties the agent encounters, the more information it stores about its environment, and it becomes better at adapting to new situations. Having a higher success rate while training also permits a safer approach to cope with novelties. Our method shows better total convergence to a threshold success rate, which is improving with the training.

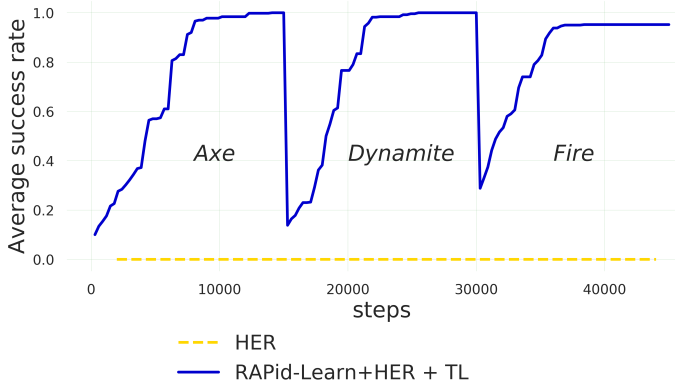
The knowledge acquired on previous novelties helps learning new novelties faster. However, it is alone not sufficient to help it to converge to a success rate of 0.9 or higher. In order to do so, it is important for the agent to further explore the environment. Therefore, the expected jump-start



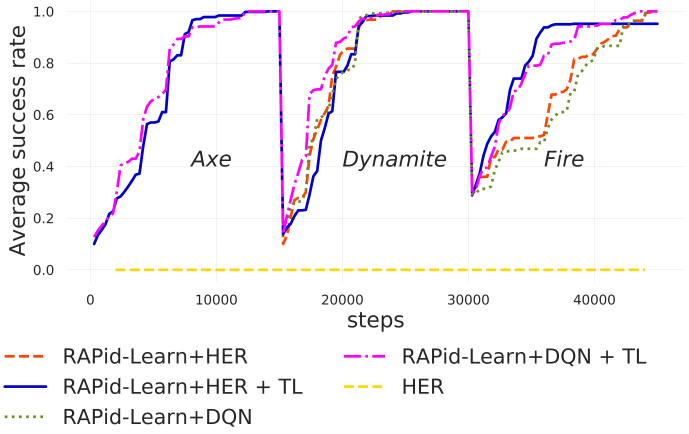
(a) RAPid-Learn+DQN *versus* RAPid-Learn+DQN+TL



(b) RAPid-Learn+HER *versus* RAPid-Learn+HER+TL (our method).

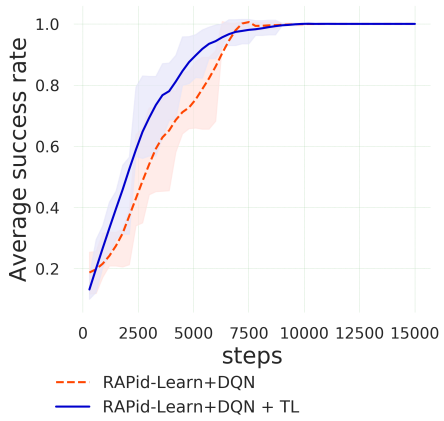


(c) HER *versus* RAPid-Learn+HER+TL (our method).

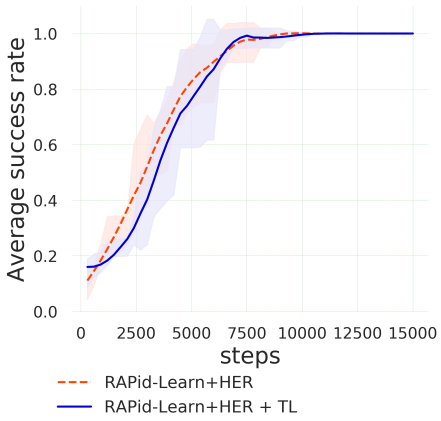


(d) HER, RAPid-Learn+HER, RAPid-Learn+DQN, and RAPid-Learn+DQN+TL *versus* RAPid-Learn+HER+TL (our method).

Figure 8: The figure shows the continual novelty injections and compares the performances of the our method versus baselines.

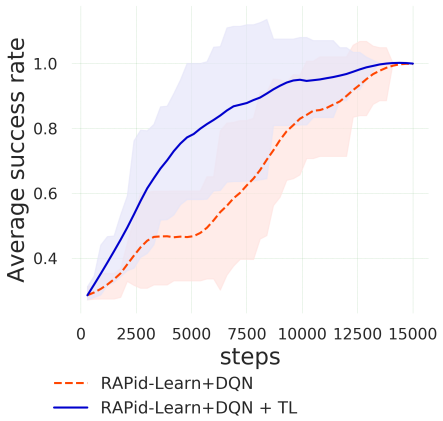


(a) Dynamite:
RAPid-Learn+DQN
& RAPid-Learn+DQN+TL

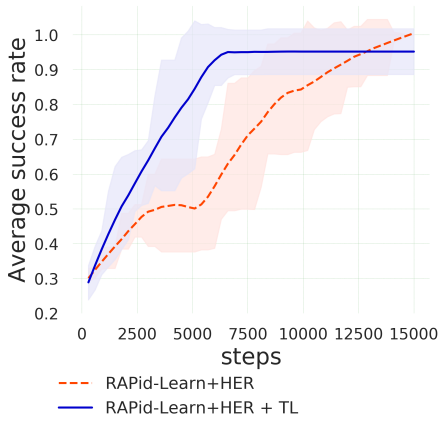


(b) Dynamite:
RAPid-Learn+HER
& RAPid-Learn+HER+TL

Figure 9: The plots show the learning curves of the operators for *dynamite* novelty and *fire* novelty. (a) shows the comparison (RAPid-Learn+DQN) and (RAPid-Learn+DQN+TL), transfer is performed from *axe* novelty. (b) shows the *dynamite* novelty comparison of HER versus our approach (RAPid-Learn+HER+TL); transfer being done from *axe* novelty.



(a) Fire:
RAPid-Learn+DQN
& RAPid-Learn+DQN+TL



(b) Fire:
RAPid-Learn+HER
& RAPid-Learn+HER+TL

Figure 10: The plots show the learning curves of the operators for *fire* novelty (a) shows the *fire* novelty performance of RAPid-Learn+DQN versus RAPid-Learn+DQN+TL; transfer is done from *dynamite* novelty (b) shows the comparison between our method (RAPid-Learn+HER+TL) and HER on the *fire* novelty; transfer is done from *dynamite* novelty.

exists yet remains tiny as the novelty policy learning still starts with a high exploration rate to learn the new goal.

We also compare our method with a pure learning methodology (shown in Figure 8(c)). It clearly showcases, that the pure learning methods fail entirely to solve such complex tasks and are unable to cope up with novelties in their environments.

Limitations & Future work

Our approach comes with limitations and certain challenges to be addressed. We highlight some current limitations and ways to extend this work in interesting directions. Automating the setting of novelty hyper-parameters seems very complex because the agent always lacks information about it. Even if HER capitalizes on information from previous trials, catastrophic forgetting can occur. This method also requires anticipating placeholders in the goal set for future novelties, and we are working on approaches that can automatically expand the learning network by transfer learning approaches. In order to showcase the viability of the method to deal with future novelties, we are working on evaluating this approach on more complex tasks and more novelty injections. A future direction is to evaluate this approach in robotics domains with continuous action spaces and stochastic environments.

Policy learning and its transfer over time can lead to expansion of the network, mainly using techniques like HER can increase the size of the replay buffer with unnecessary information. This can be a problem, especially in lifelong learning scenarios. We are working on extensions that can learn functions to filter the replay buffer (Czarnecki et al. 2019) and prevent catastrophic forgetting. Human-robot collaboration is crucial in solving complex multi-step tasks and accelerating the agent's learning to overcome novelties. HER has already been studied for this purpose and has shown improvement in learning by obtaining information from human demonstrations (Nair et al. 2017). While our goal is to automate machines in their general behavior further, we also plan to demonstrate the potential of this approach for human-robot interaction.

We plan to integrate this technique into real-world industrial projects so that industrial machines can automatically plan tasks and adapt to novelties efficiently. This will require understanding the physical properties of novelties to assimilate them into a simulated environment. Moreover, learning in such scenarios should be extremely sample-efficient while prioritizing safety. In this approach, we attempt to address the latter part, and we are working on ideas to address the former limitation for real-world integration. To this end we plan on performing significantly more inferences on the fly to restrict learning through experimentation. We also plan on extending this method to robotics scenarios with continuous actions, where in prior work we have developed methods for action discovery using segmentation (Gizzi, Castro, and Sinapov 2019) and behavior babbling (Gizzi et al. 2021) but stopped short of using RL to learn action executors when facing novelties.

Conclusion

In this paper, we showed that an integrated approach of planning and learning, using HER and policy transfer, allows the agent to gain knowledge from previous trials. This approach provides a solution for dealing with novelties in a way that is beneficial to the agent by improving its future performance on other novelties and continuously increasing the agent's learning efficiency. We evaluated our approach on an extremely complex task scenario which was injected with three novelties, and compared it with an integrated planning and learning approach with and without transfer learning.

Acknowledgements

This work was funded in part by DARPA contract W911NF-20-2-0006.

References

Abel, D.; Jinnai, Y.; Guo, S. Y.; Konidaris, G.; and Littman, M. 2018. Policy and value transfer in lifelong reinforcement learning. In *ICML*, 20–29. PMLR.

Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; and Ribas, R. e. a. 2019. Solving rubik's cube with a robot hand. *arXiv:1910.07113, 2019.*

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, P.; and Zaremba, W. 2017. Hindsight Experience Replay. *CoRR*, abs/1707.01495.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Cheung, W. C.; Simchi-Levi, D.; and Zhu, R. 2020. Reinforcement learning for non-stationary markov decision processes: The blessing of (more) optimism. In *ICML*, 1843–1854. PMLR.

Czarnecki, W. M.; Pascanu, R.; Osindero, S.; Jayakumar, S.; Swirszcz, G.; and Jaderberg, M. 2019. Distilling Policy Distillation. In Chaudhuri, K.; and Sugiyama, M., eds., *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, 1331–1340. PMLR.

Ghallab, M.; Nau, D.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

Gizzi, E.; Castro, M. G.; and Sinapov, J. 2019. Creative problem solving by robots using action primitive discovery. In *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 228–233. IEEE.

Gizzi, E.; Hassan, A.; Lin, W. W.; Rhea, K.; and Sinapov, J. 2021. Toward creative problem solving agents: Action discovery through behavior babbling. In *2021 IEEE International Conference on Development and Learning (ICDL)*, 1–7. IEEE.

Goel, S.; Shukla, Y.; Sarathy, V.; Scheutz, M.; and Sinapov, J. 2022. RAPid-Learn: A Framework for Learning to Recover for Handling Novelties in Open-World Environments.

Goel, S.; Tatiya, G.; Scheutz, M.; and Sinapov, J. 2021. NovelGridworlds: A Benchmark Environment for Detecting and Adapting to Novelties in Open Worlds. In *AAMAS Adaptive Learning Agents (ALA) Workshop*.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating"Ignoring Delete Lists"to Numeric State Variables. *Journal of artificial intelligence research*, 20: 291–341.

Khetarpal, K.; Riemer, M.; and amd Doina Precup, I. R. 2020. Towards Continual Reinforcement Learning: A Review and Perspectives. *arXiv:2012.13490v1*.

Lecarpentier, E.; Abel, D.; Asadi, K.; Jinnai, Y.; Rachelson, E.; and Littman, M. L. 2020. Lipschitz lifelong reinforcement learning. *arXiv preprint arXiv:2001.05411*.

Leike, J.; Martic, M.; Krakovna, V.; Ortega, P. A.; Everitt, T.; Lefrancq, A.; Orseau, L.; and Legg, S. 2017. AI safety gridworlds. *arXiv preprint arXiv:1711.09883*.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL-the planning domain definition language.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5602.

Muhammad, F.; Sarathy, V.; Tatiya, G.; Goel, S.; Gyawali, S.; Guaman, M.; Sinapov, J.; and Scheutz, M. 2021. A Novelty-Centric Agent Architecture for Changing Worlds. In *Proceedings of 20th International Conference on Autonomous Agents and Multiagent Systems*.

Nair, A.; McGrew, B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2017. Overcoming Exploration in Reinforcement Learning with Demonstrations. *CoRR*, abs/1709.10089.

Riemer, M.; Cases, I.; Ajemian, R.; Liu, M.; Rish, I.; Tu, Y.; and Tesauro, G. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *Seventh International Conference on Learning Representations*.

Sarathy, V.; Kasenberg, D.; Goel, S.; Sinapov, J.; and Scheutz, M. 2021. SPOTTER: Extending Symbolic Planning Operators through Targeted Reinforcement Learning. In *AAMAS-21*, 1118–1126.

Sarathy, V.; and Scheutz, M. 2018. MacGyver problems: Ai challenges for testing resourcefulness and creativity. *Advances in Cognitive Systems*, 6: 31–44.

Schaul, T.; Horgan, D.; Gregor, K.; and Silver, D. 2015. Universal Value Function Approximators. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 1312–1320. Lille, France: PMLR.

Schmidhuber, J. 2013. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *Frontiers in psychology*, 4:313. doi: 10.3389/fpsyg.2013.00313.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1): 181–211.