Build generally reusable agent-environment interaction models

Jun Jin[1,*] **Hongming Zhang**[1,2] **Jun Luo**[1]
[1]Huawei Noah's Ark Lab [2]University of Alberta

Abstract

This paper tackles the problem of how to pre-train a model and make it generally reusable backbones for downstream task learning. In pre-training, we propose a method that builds an agent-environment interaction model by learning domain invariant successor features from the agent's vast experiences covering various tasks, then discretize them into behavior prototypes which result in an embodied set structure. To make the model generally reusable for downstream task learning, we propose (1) embodied feature projection that retains previous knowledge by projecting the new task's observation-action pair to the embodied set structure and (2) projected Bellman updates which add learning plasticity for the new task setting. We provide preliminary results that show downstream task learning based on a pre-trained embodied set structure can handle unseen changes in task objectives, environmental dynamics and sensor modalities.

1 Introduction

Can we pre-train a generally reusable backbone to accelerate downstream task reinforcement learning (RL)? While such backbones are popular in computer vision and natural language processing (e.g., pre-trained ResNet (He et al., 2016) and pre-trained GPT3 (Brown et al., 2020), pre-trained decision-making backbones that support general downstream task learning are a few. The challenges largely attribute to the vast complexity and variety of decision-making tasks, that the pre-trained model will suffer from the out-of-distribution nightmare when reused to learn a new task. For example, a downstream task's environment dynamics or task objectives may be unseen during pre-training. Or we may need to add an extra sensor modality (e.g., language) to the agent to best fit the task requirement (e.g., understanding human commands). How can we pre-train a backbone that handles the above-unseen changes in downstream task learning, without making any assumptions on the task similarity (Duan et al., 2016; Gupta et al., 2018) of environment similarity (Packer et al., 2018; Rajeswaran et al., 2016)?

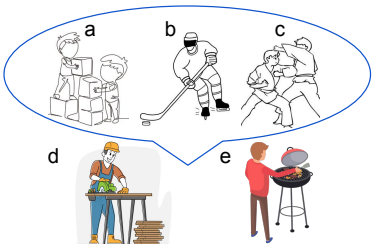

Figure 1: What models are generally shared between heterogeneous tasks that can be reused to accelerate new task learning in unseen settings?

Empirically, humans are capable of reusing experience from one task to another, and the tasks we can handle are surprisingly heterogeneous (Fig. 1), from playing blocks, hockey and karate, to learning a carpenter or cooking job. It's natural to ask, (1) what model is generally shared between heterogeneous tasks? And (2) how can it fit in any downstream task learning with unseen settings?

Systematically, thinking on the above questions results in two perspectives of viewing what component is shared across tasks—the world and the agent (Harutyunyan, 2020). The first one seeks to build

*Correspondence to jun.jin1@huawei.com

Foundation Models for Decision Making Workshop at Neural Information Processing Systems, 2022.

a *Bayesian world model* (Ha & Schmidhuber, 2018), with which, the agent stores prior knowledge about the physical world. For example, "balls will roll" and "walking on ice is slippery". Then the Bayesian agent selects a proper world configuration for downstream task learning. However, since downstream tasks can be complex and diverse, then how large and how accurate the world model should be is not clear. The second perspective seeks to build an *embedded agency* (Orseau & Ring, 2012), that the agent gradually constructs knowledge learned from its past experience, respecting the fact that the world is far more complex than the agent. The constructed knowledge can be representations or models and will accelerate downstream task learning. The *embedded agency* perspective can also gradually build a world model for the agent, but unlike the *Bayesian world model*, the agent's past experience shapes what the world model is, how large it is, and how accurate it is. This paper relates to the second viewpoint.

Though the *embedded agency* view is conceptually appealing, technically how to build it and the promising benefits w.r.t. sample efficiency or continual learning capability are not clear. Without being too ambitious to build a complete embedded agency, in this paper, we focus on a more practical starter: *how the agent constructs a model from its embodied experience, and how it accelerates new task learning when environments, task objectives, and even sensor modalities are unseen*.

Specifically, we format the problem of building a generally reusable model as an embodied set construction approach that constructs a domain invariant set structure using a large dataset covering the agent's vast task experiences. We call it an agent-environment interaction model since the set elements are discretized successor features (Barreto et al., 2017) which represent how a decision-making agent interacts with the world. We call our approach embodied set construction since the set structure is constructed so that it is a shared component across tasks and environments, which can be viewed as an embodiment of the agent. We call it generally reusable since learning by projecting onto the embodied set structure enables learning stability-plasticity. Our contributions are as follows:

- We propose an *embodied set construction* approach to pre-train a reusable agent-environment interaction model by firstly learning domain invariant successor features from the agent's past experience, then discretizing them to behaviour prototypes which result in an embodied set structure. Note that the constructed embodied set structure, though optimized to be domain-invariant, still suffers the out-of-distribution (O.O.D.) problem when reused in downstream tasks under unseen settings.
- To make the pre-trained model generally reusable, we propose two projection-based techniques that enable learning stability-plasticity: (1) *embodied feature projection* which retains previous knowledge by projecting the new task's observation-action pairs onto the embodied set structure, and (2) *projected Bellman update* which adds plasticity for the new task learning.
- Combined together, we propose a complete solution of pre-training a generally reusable model and plugging it as a backbone to accelerate downstream task learning.

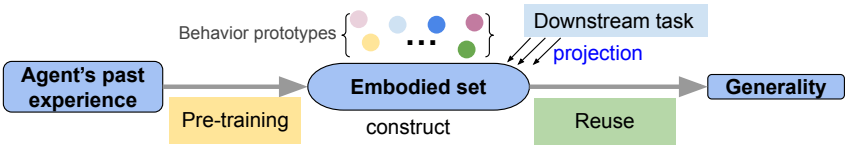

Figure 2: Diagram of our proposed method.

Our experiments show that downstream task learning based on a pre-trained embodied set structure can handle unseen changes of task objectives, environmental dynamics and sensor modalities, without assuming any environmental or task distribution priors.

2 Related work

Pre-training a backbone to accelerate reinforcement learning is not new. We classify existing works into four categories depending on what is reused in downstream task learning.

(1) *Reusable representations*: Most existing works fall into this category that aims to learn a representation either to improve sample efficiency or task generalization. Methods include task-agnostic ones,

for example, using VAE (Kingma & Welling, 2013) and a pre-trained ResNet (Shah & Kumar, 2021) for state representations), self-supervised representation learning methods (RAD (Laskin et al., 2020), CURL (Srinivas et al., 2020) , CPC (Oord et al., 2018), and ATC (Stooke et al., 2021)) for image states, and task-specific ones(Jin et al., 2022), for example, using a behaviour similarity measure (Bi-simulation metrics (Zhang et al., 2020)) to constrain the representation which results in better generalization performance.

(2) *Reusable policy/value functions*: This approach aims to pre-train a policy or value function that can be further fine-tuned in downstream task learning. Methods include assuming task distribution priors (meta-RL (Duan et al., 2016)), assuming environmental distribution priors (generalizable RL (Packer et al., 2018) , robust RL (Rajeswaran et al., 2016)), and contextual MDP based ones (Huang et al., 2020). Recently, representing a policy using generative sequential models has gained increasing popularity (Gato (Reed et al., 2022), Decision Transformers (Chen et al., 2021)). However, their capability of downstream task learning is commonly restricted to the training setup, and handling large changes when reused remains challenging (Reed et al., 2022).

(3) *Reusable skills and options*: This approach assumes tasks can be decomposed into shareable skills (Sutton et al., 2011) that can be learned via time abstraction—learning options as skills. The difficulty lies in how to discover reusable skills, and it relates to recently proposed works in the *unsupervised RL* literature (Laskin et al., 2022). Current methods for skill discovery include curiosity and diversity-based ones (intrinsic controls (Gregor et al., 2016), DIYAN (Eysenbach et al., 2018), behaviour basis-based ones (Proto-value functions (Mahadevan, 2005), and variational intrinsic successor features (Hansen et al., 2019).

3 Methodology

In the below discussions, we use the Partial Markov Decision Process (POMDP) formation since this setting fits in most real-world reinforcement learning tasks where the full state is not obtainable and the agent is only given observations from its onboard sensors (Igl et al., 2018). A POMDP is defined as: $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{T}, r, \gamma)$, where $\mathcal{X}$ denotes the observation space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the environment transition dynamics, r is the reward function and γ denotes the discount factor.

3.1 Reusable agent-environment interaction models

(1) Embedded agency: small agent, big world Unlike the *Bayesian world model* approach (Ha & Schmidhuber, 2018) that aims to construct a big model for the agent as a base knowledge about the world, the *Embedded agency*(Orseau & Ring, 2012) approach takes the perspective on how the small agent perceives, controls and gradually grows based on its own experience. Unlike a *Bayesian world model* where states and models are objective, an *embedded agency* approach is subjective(Harutyunyan, 2020) that agent's past experience will shape its behaviors in new tasks. This idea of intelligence closely relates to Descarte's theory of representationalism and mind-body dualism, and to the modern theory of embodied embedded cognition(Haugeland, 1993) in the philosophy of mind research. The *embedded agency* approach is appealing in that it promises a scalable learning architecture to build a world model based on the agent's experience in a self-improvement manner, however, it is also ambitious. This paper investigates the core idea of *embedded agency* about how an agent's past experience can shape its new task learning in a pre-training and reusing paradigm.

(2) On agent-environment boundary Conceptually, from the *embedded agency* perspective, a generally reusable model in heterogeneous environments and tasks should be the agent itself, since the agent is the only shared component across domains. Practically, using the agent's past experience to extract the agent model is difficult since the agent-environment boundary is not the same as we think in a physical system, like a robot. (Jiang, 2019) provide a detailed analysis of this problem. (Sutton & Barto, 2018) in chapter 3.1 explains that the agent-environment boundary is often task-specific in different abstraction levels and the boundary could change across tasks and environments, which makes exacting a general agent model from its various task data difficult.

Instead of extracting an agent model, we propose to extract a general agent-environment interaction model that is commonly shared across tasks and environments, and thus can be viewed as an embodiment of the agent. This proposal is explained below.

(3) Generally reusable agent-environment interaction model In pre-training, suppose the agent's past experiences $\mathcal{D}$ are generated from M different environments and collected using unlimited behaviour policies induced by unlimited reward functions. This will compose a set of different POMDPs: $\{\mathcal{M}_i\}_{i=1}^M$, where $\mathcal{M}_i = (\mathcal{X}_i, \mathcal{A}, \mathcal{T}_i, \gamma)$ is a reward-free POMDP that drops the task reward for simplicity. $\mathcal{A}$ without a subscript i means the agent's action space is unchanged across environments since the embedded agency considers the same agent across environments and tasks. Let's assign a one-hot vector encoding Y_i as the label for each environment domain $\mathcal{M}_i$, then we have a set of domain labels $\mathcal{Y} = \{Y_i\}_{i=1}^M$, the collected dataset for pre-training can be denoted as $\mathcal{D} = \{D_{Y_i}\}_{i=1}^M$, where D_{Y_i} is the dataset collection in environment domain Y_i.

We use successor features (SF)(Barreto et al., 2017) to capture the agent-environment interaction model since SFs summarize the dynamics induced by the environment when following a behaviour policy. The first step of building a generally reusable agent-environment interaction model is to learn cross-domain transferable successor features in order to extract a general agent-environment interaction model. For simplicity, we consider a random uniform behaviour policy π_0 for all environment domains. Recall that, SF is defined as the expected cumulative features of ϕ by following π_0 starting at a specific state. The SF of (x^i, a) in environment domain Y_i is defined as:

$$\psi^{\pi_0,i}(x^i, a) = \mathbb{E}^{\pi_0}_{(x,a,x')\sim\mathcal{T}_i}[\sum_{t'=t}^{\infty} \gamma^{t'-t}\phi_{x_{t+1}}|X_t = x^i, A_t = a]. \tag{1}$$

, which satisfies the Bellman Equation as below:

$$\psi^{\pi_0,i}(x^i, a) = \phi_{x^i} + \gamma\mathbb{E}^{\pi_0}_{(x,a,x')\sim\mathcal{T}_i}[\psi^{\pi_0,i}(x^i_{t+1}, a_{t+1})|X_t = x^i, A_t = a] \tag{2}$$

, and it can be learned by minimizing the temporal difference (TD) error:

$$\delta^2_{sf,Y_i} = \left\|\phi_{x^i_t} + \gamma\psi^{\pi_0,i}(x^i_{t+1}, a_{t+1}) - \psi^{\pi_0,i}(x^i_t, a_t)\right\| \tag{3}$$

Next, let's consider a set of domains $\mathcal{Y} = \{Y_i\}_{i=0}^M$. The goal is to learn a successor feature approximation function $f(.;\theta_{sf}) : \mathcal{X}_i \times \mathcal{A} \rightarrow \boldsymbol{\psi}$ with parameters θ_{sf} that is transferable across all the domains $\mathcal{Y}$. Inspired by (Feng et al., 2019), we add two constraints to Eq. 3 in order to make the learned SF cross-domain transferable, using the mutual information definition $I(.)$:

$$\begin{aligned} min \quad & \mathcal{L}_{sf} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{(x,a,x')\sim\mathcal{D}_{Y_i}}[\delta^2_{sf,Y_i}] \\ s.t. \quad & I(\psi^{\pi_0}, Y) < \epsilon_u; \quad I(\psi^{\pi_0,i}, x^i) > \epsilon_l, \quad \forall i \in \{1, ..., M\} \end{aligned} \tag{4}$$

In the first constraint, ψ^{π_0} is the SF for an arbitrary environment domain, and Y is the corresponding domain label. It limits the mutual information between an SF and its domain label to a threshold ϵ_u in order to make the learned successor feature domain-invariant. The domain index Y_i is dropped since it generally applies to all domains. The second constraint term maintains the mutual information between an SF and its input domain-specific observations[2] above a threshold ϵ_l, in order to prevent the optimization from collapsing to a trivial solution, for example, a random feature space is also domain-invariant but does not include any useful information for task learning. Technical details about optimizing Eq. 4 are shown in Sec. 3.2.

Note that, learning a cross-domain transferable successor feature representation *will not solve* the out-of-distribution (O.O.D.) problem when reusing it in downstream tasks with unseen changes, so that directly plugin the pre-trained successor features, as a common approach in ((Barreto et al., 2017, 2020)), **will not make our pre-trained model generally reusable**. We propose two techniques to tackle this problem, *(1) embodied set construction* that discretizes the successor features into prototype sets (Sec. 3.2); *(2) feature projection* and *projected Bellman updates* to enable learning stability-plasticity (Sec. 3.3). Combined together, we make the pre-trained agent-environment interaction model generally reusable.

3.2 Pre-training: embodied set construction

The pre-training process is proposed as an embodied set construction method that has two steps:

[2]We use the domain index Y_i since it requires to estimating a specific domain's marginal distribution $p^i(x)$ that $x^i \sim p^i(x)$ during training, as detailed in Appendix. A.1.

(1) Learn cross-domain transferable successor features Let a neural network parameterized with θ_{sf} to approximate the SF representation: $\psi^{\pi_0}(x, a; \theta_{sf})$, where (x, a) comes from an arbitrary environment domain. Our aim is to use loss Eq. 4 to find the optimal θ_{sf}. By adding the Lagrangian multipliers λ_u and λ_l, the Lagrangian dual of Eq. 4 is:

$$\min_{\theta_{sf}} \quad \mathcal{L}_{sf} + \lambda_u I(\psi^{\pi_0}, Y) - \lambda_l \sum_{i=1}^{M} I(\psi^{\pi_0,i}, x^i) \quad (5)$$

Directly optimizing the mutual information (MI) terms of Eq. 13 in high-dimensional space is challenging, we provide tractable solutions by approximating the upper bound and lower bound, as detailed in Appendix. A.1. Optimizing Eq. 13 will result in transferable $\psi^{\pi_0}(x, a; \theta_{sf})$.

Algorithm 1: Embodied Set Construction

Input: Offline dataset $\mathcal{D} = \{D_{Y_i}\}_{i=1}^{M}$, $\psi^{\pi_0}(x, a; \theta_{sf})$, embodied set size N
Result: Embodied set structure Ω^e
Initialize an empty embodied set $\Omega^e = \{\}$
Initialize an empty successor feature vector list $\mathbf{L}_{sf} = \{\}$
$\mathcal{D} \leftarrow$ Shuffle $(\mathcal{D})$
for *each* $(x, a, x') \in \mathcal{D}$ **do**
  // Compute cross-domain transferable successor features
  $\mathbf{L}_{sf} \leftarrow$ Append $\psi^{\pi_0}(x, a; \theta_{sf})$
end
// Constructing embodied agent state set
K-means clustering $(\mathbf{L}_{sf}, N)$
$\Omega^e = \{\mathbf{e}_i\}_{i=1}^{N}$ = cluster-centers as behavior prototypes

(2) Discretize to behavior prototypes: construct an embodied set structure To facilitate downstream task learning with our proposed *embodied feature projection* and *projected Bellman updates* (Sec. 3.3), we discretize the learned cross-domain successor features in three steps: (i) Compute all the successor features using the learned $\psi^{\pi_0}(x, a; \theta_{sf})$ from $\mathcal{D}$ containing all the environment domain samples. (ii) Cluster all the successor features using mini-batch K means. (iii) Collect the center of each cluster to construct a set structure Ω^e. This is summarized in Algorithm 1.

We call Ω^e as *embodied set* since it is the component shared across environments and tasks that can be viewed as the embodiment of the agent. And each $\mathbf{e}_i \in \Omega^e$ is called a prototype since it is the center of a cluster that represents a prototype agent-environment interaction behaviour.

3.3 Re-use: a backbone for general downstream task learning

Let a downstream task denoted as $\mathcal{M}_u = (\mathcal{X}^u, \mathcal{A}, \mathcal{T}^u, r^u, \gamma)$, where the observation space $\mathcal{X}^u$, dynamics $\mathcal{T}^u$, and task objectives r^u can be unseen in pre-training. The action space $\mathcal{A}$ remains the same since we assume the same agent learning a new task based on its past experience.

Reusing the pre-trained model under the above heterogeneous settings is difficult due to out-of-distribution concerns. Traditional methods will not work. For example, in methods reusing the pre-trained representation (Shah & Kumar, 2021; Kingma & Welling, 2013), changes in the environment will cause non-stationarity in the observation space that directly plugs in the learned representation will make the learning even worse than learning from scratch. For another example, in methods reusing the pre-trained successor features to fast compute the value functions (Barreto et al., 2020), changes in task objective or the environment will break the linear reward feature assumption.

We propose to avoid directly plugging in the pre-trained model, but to use the pre-trained model—embodied set Ω^e as a base structure for projection-based techniques to accelerate the downstream task learning by tackling learning stability and plasticity explicitly.

(1) Stability: Retain previous knowledge by embodied feature projection We define a feature projection operator $\Pi_{\Omega^e}(x, a)$, which takes the input of (x, a), localizes its projection on the constructed embodied set Ω^e, and returns the localized feature vector $\mathbf{e}$:

$$\Pi_{\Omega^e}(x, a) = \mathbf{e}, \quad \forall \mathbf{e} \in \Omega^e, \text{ find the smallest } \xi(x, a, \mathbf{e}) \quad (6)$$

, where $\xi(x, a, \mathbf{e}) = \|\psi^{\pi_0}(x, a; \theta_{sf}) - \mathbf{e}\|$, or it can be any other distance metric functions. The feature projection operator $\Pi_{\Omega^e}(x, a)$ retains previous knowledge by always matching the new task experience with the closest prototype $\mathbf{e}$ in the embodied set.

Note that $\Pi_{\Omega^e}(x, a)$ also works for changed sensor modalities. For example, assuming a task requires the agent to understand textual commands, then adding an extra-textual observation z will augment the observation to $[x, z]$. The feature projection operator $\Pi_{\Omega^e}(x, a)$ still applies here by using the unchanged sensory modality part x. We will show that in Sec. 5.1 (3).

(2) Plasticity: Adapt to changes by projected Bellman Updates We use the below projected Bellman updates to accelerate learning the unknown downstream task $\mathcal{M}_u = (\mathcal{X}^u, \mathcal{A}, \mathcal{T}^u, r^u, \gamma)$:

$$Q^\pi(x, a) = \mathbb{E}_{(x,a,x')\sim\mathcal{T}^u}[r^u + \gamma V^\pi_{proj}(\Pi_{\Omega^e}(x', \arg\max_{a'} Q^\pi(x', a'))] \tag{7}$$

$$V^\pi_{proj}(\Pi_{\Omega^e}(x, a)) = \mathbb{E}_{(x,a,x')\sim\mathcal{T}^u}[r^u + \gamma \max_{a'} Q^\pi(x', a') \tag{8}$$

We maintain two value functions—the task Q^π and a projected version V^π_{proj}, to support each other's learning in a bidirectional improvement manner. The motivation is that the projected function V^π_{proj} can learn faster than Q^π since it is defined on the pre-trained set Ω^e that retains past experience, but is not accurate since Ω^e does not adapt to the new task setting. Meanwhile, the task Q-value function Q^π should be more accurate but will take a longer time if learned from scratch. Learning that alternates Bellman updates Eq. 7 and 8 will play a trade-off between retaining the previous knowledge or adapting to the new tasks.

(3) Reuse example: DQN-embodied While the above two techniques *embodied feature projection* and *projected Bellman updates* can generally apply to any RL methods in learning a new task, we use DQN as an example to show how to use them to accelerate downstream task learning. Assume we use a neural network parameterized with θ_u to approximate Q^π, and another neural network parameterized with $\mathbf{w}_u$ to approximate V^π_{proj}. According to Eq. 7 and 8, we compute the target value for Q^π and V^π_{proj} at training iteration i as:

$$y_i = \mathbb{E}_{(x,a,x')\sim\mathcal{T}^u}[r^u + \gamma V^\pi_{proj}(\Pi_{\Omega^e}(x', \arg\max_{a'} Q(x', a'; \theta_{u,i-1})); \mathbf{w}_{u,i-1})] \tag{9}$$

$$y_{proj,i} = \mathbb{E}_{(x,a,x')\sim\mathcal{T}^u}[r^u + \gamma \max_{a'} Q^\pi(x', a'; \theta_{u,i-1})] \tag{10}$$

Then, the learning objectives formulated as an LMSE loss can be written as:

$$\mathcal{L}_i(\theta_{u,i}) = \mathbb{E}_{(x,a,x')\sim\mathcal{T}^u}[(y_i - Q^\pi(x, a; \theta_{u,i}))^2] \tag{11}$$

$$\mathcal{L}_i(\mathbf{w}_{u,i}) = \mathbb{E}_{(x,a,x')\sim\mathcal{T}^u}[(y_{proj,i} - V^\pi_{proj}(\Pi_{\Omega^e}(x, a)); \mathbf{w}_{u,i})^2] \tag{12}$$

During training, we alternate learning Q^π and V^π_{proj}. A full description of the proposed DQN-embodied algorithm is summarized in Appendix A.2.

4 Experimental setup

We aim to test *the generality* of our method by increasing the difference w.r.t. the environment, the task objective, and the agent's sensory modality, between pre-training and reuse.

(1) Choosing the environment To serve the above evaluation purpose, we require the evaluation environment should be easy to change the task objective (reward), environmental dynamics, and sensor modalities. We choose MiniGrid (Chevalier-Boisvert et al., 2018) since it is highly re-configurable and easy to reproduce results. We test tasks including "goal reaching", "picking up the grey box" and "putting the green key near the yellow ball".

We use the default partial observability setting that the agent can only observe nearby 7×7 cells. Each cell is described with 3 input values: [OBJECT_IDX, COLOR_IDX, STATE], which gives the object id, cell color and door state (open, closed, locked). Therefore, the observation x is a $7 \times 7 \times 3$ dimensional vector. In changing the sensor modality test, we manually add a textual observation by using the embedding of a text z_t to augment x_t with $[x_t, z_t]$.

The agent's action space remains unchanged during pre-training and downstream task learning, which is a discrete set containing 6 action choices: turn left, turn right, move forward, pick up an object, drop the object being carried, and toggle (open doors, interact with objects).

(2) Dataset collection We consider 15 different room layouts environments during pre-training, specifically, 1 layout using 1 room, 4 layouts using 2 rooms and 10 layouts using 4 rooms. The room layouts are randomly generated. To fully utilize the agent's action capability, especially for "picking up an object", "dropping an object being carried", and "toggle", we randomly place objects (blue keys, blue doors, and different colored balls). For each room layout, we randomly configure 10 different object placement scenarios. Therefore, we have 150 environments to generate the agent's experience

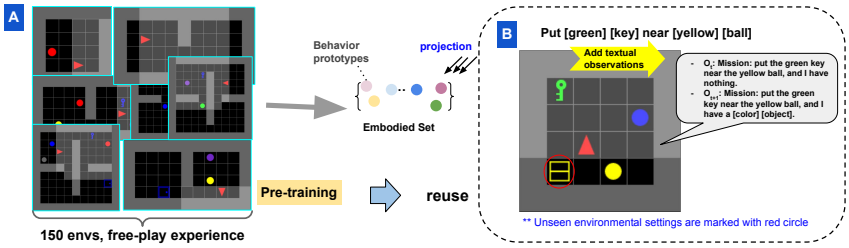

Figure 3: **A:** Environments for collecting the pre-training dataset with different room layouts and random objects (blue keys, blue doors, coloured balls) for the agent to interact with. **B:** Downstream task "putting the green key near the yellow ball" uses an unseen environment (green key, coloured boxes unseen during pre-training), unseen task objective, and unseen sensor modality by adding textual observations. Note this task requires text prompts otherwise it is difficult to solve by a vanilla RL agent(Chevalier-Boisvert et al., 2018)

for pre-training. We then let the agent explore uniformly to navigate the rooms and interact with the objects without any task goals to collect samples. Therefore, we have 150 task-free POMDPs $\{\mathcal{M}_i\}_{i=1}^{150}$, which result in 150 environment domains $\{Y_i\}_{i=1}^{150}$, with each Y_i is a one-hot encoding vector to denote the domain label.

In summary, during data collection, we created 150 play zones for the agent to explore freely without defining any task goals. For each domain, we collected $1M$ samples which compose the dataset $\mathcal{D} = \{\mathcal{D}_i\}_i^{150}$ with $150M$ samples. Each samples is a tuple (x, a, x').

(3) Pre-training: embodied set construction Following (Hoang et al., 2021), we pre-train a variational auto-encoder (VAE (Kingma & Welling, 2013)) that maps observation x_t to a feature vector ϕ_t, which is then used to learn the cross-domain transferable successor features by learning the function $\psi^{\pi_0}(x, a; \theta_{sf})$ using methods proposed in Sec. 3.2. During embodied set construction, we set $N = 10K$ to discretizes all SFs into 10K clusters and construct the embodied set Ω^e representing 10K prototypes behaviors.

(4)Re-use: a backbone for downstream task learning Given the pre-trained embodied set structure Ω^e, we test how it can be reused to accelerate downstream task learning using our proposed DQN-embodied method (see Algorithm 2), as described below.

5 Preliminary results

We conduct the *generality test* by adding differences between pre-training and reuse. Fig. 3 gives an overview of task setup in the below three tests.

Unseen task objective + unseen environment + unseen sensor modalities We test the generality of our proposed method by adding an unseen task objective, environment settings, and sensor modality in downstream task learning. The task is "putting the green key near the yellow ball" wherein the agent needs to understand "green key" (unseen in pre-training), "yellow ball" and "near". The task requires the agent to find the green key, carry it and drop the key in a cell nearby the yellow ball. The reward function is defined as $r(x, a, x') = 0.1$ (if pick up the green key), 10.0 (if succeed).

Adding textual sensor modality: Note that this task requires text prompts otherwise is difficult to solve. We manually design three types of textual observations for the agent— The mission statement "Put the green key near the yellow ball" + different textual statuses: {"and I have nothing", "and I have the [color] [object]"} depends on what the agent is carrying. The textual observation it then converted to an embedding vector o using a pre-trained sentence transformer MiniLM (384 dimensional output, 80 MB model size) (Wang et al., 2020) using a libary provided by (Reimers & Gurevych, 2019). Therefore, the agent's observation at time t is augmented as : $[x_t, o_t]$, where $x_t \in \mathbb{R}^{7\times7\times3}$ is the same observation as in pre-training, and $o_t \in \mathbb{R}^{384}$ is the extra sensor modality added to the agent to provide textual observations.

Baselines: We compare our method (*DQN-embodied*) to the following baselines. (1) Without pre-training: vanilla DQN (Mnih et al., 2013); (2) Reuse the pre-training dataset: experience replay (ER) which reuses the whole dataset $\mathcal{D}$ in the buffer to alleviate the forgetting issue (Rolnick et al., 2019); (3) Reuse a pre-trained backbone: DQN-VAE (Kingma & Welling, 2013) is used as a backbone for

downstream task learning; (4) Without textual observations (DQN-no-text) using a vanilla DQN (Mnih et al., 2013). Note that baselines DQN-VAE can not directly handle an extra sensor modality. When taking the o_t as inputs to the above baselines, we concatenate the VAE encoded observation with o_t.

We test the downstream task learning performance of all methods by evaluating the success rate every 10K training steps using 100 independent runs with different random seeds. Results (Fig. 4) show that (1) Without textual observations, a vanilla DQN agent can not solve the task; (2) Since the new task setting is heterogeneous compared to tasks in the training pools, directly reusing previous experience (the ER baseline) will deteriorate the downstream task learning performance (vanilla DQN is better than ER); (3) Also, since the new task is never seen during pre-training, DQN based on pre-trained VAE performs the worst. (4) For the test task that has unseen task objectives (put a color object near another color object), unseen environmental settings (box object never seen in pre-training), and unseen sensor modalities (adding textual observations), projecting the downstream task samples on the pre-trained embodied set will result in better sample efficiency than all baselines.

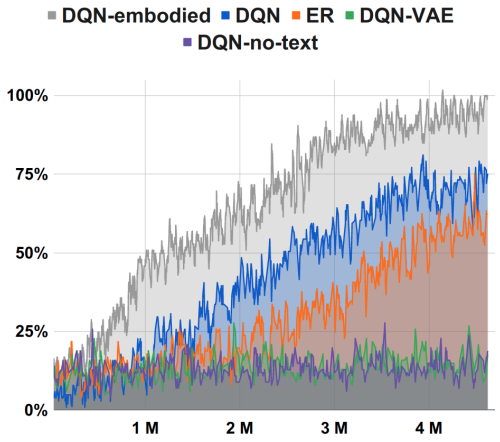

Figure 4: Preliminary results of reusing generality test. X-axis represents the sample size. Y-axis represents the success rate of 100 independent runs.

6 Conclusion

This paper tackles the problem of pre-training a generally reusable model for downstream tasks with unseen task objectives, environments, and even with unseen sensor modalities. Our approach is inspired by the *embedded agency* perspective that the agent's past experience shapes its future decision-making behaviors. We propose to learn an agent-environment interaction model by extracting cross-domain transferable successor features using the agent's various task experiences. To make the model generally reusable, we propose to discrete it into behavior prototypes, resulting in an embodied set structure that further supports *embodied feature projection* and *projected Bellman updates* to enable downstream task learning with heterogeneous settings. Preliminary results show that downstream task learning by projecting on our proposed embodied set structure can handle unseen changes in task objectives, environmental dynamics and sensor modalities.

For future works, more comprehensive experiments are needed to understand why and how the pre-trained embodied set structure can be generally reused in downstream task learning. It's also worth working on methods that can continually update the constructed *embodied set structure* given that more task experience is gained, thus forming an *embedded agency* version of continual reinforcement learning (Khetarpal et al., 2020) solution.

References

Andre Barreto, Will Dabney, Remi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Andre Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117 (48):30079–30087, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for gymnasium. https://github.com/Farama-Foundation/MiniGrid, 2018.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl 2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Zeyu Feng, Chang Xu, and Dacheng Tao. Self-supervised representation learning from multi-domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3245–3255, 2019.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in neural information processing systems*, 31, 2018.

David Ha and Jurgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019.

Anna Harutyunyan. What is an agent?, 2020. URL http://anna.harutyunyan.net/wp-content/uploads/2020/09/What_is_an_agent.pdf.

John Haugeland. *Mind embodied and embedded.* 1993.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Christopher Hoang, Sungryull Sohn, Jongwook Choi, Wilka Carvalho, and Honglak Lee. Successor feature landmarks for long-horizon goal-conditioned reinforcement learning. *ArXiv*, abs/2111.09858, 2021.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020.

Maximilian Igl, Luisa M. Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *ICML*, 2018.

Nan Jiang. On value functions and the agent-environment boundary. *ArXiv*, abs/1905.13341, 2019.

Jun Jin, Daniel Graves, Cameron Haigh, Jun Luo, and Martin Jägersand. Offline learning of counterfactual predictions for real-world robotic reinforcement learning. *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3616–3623, 2022.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *ArXiv*, abs/2012.13490, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Cic: Contrastive intrinsic control for unsupervised skill discovery. *arXiv preprint arXiv:2202.00161*, 2022.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pp. 553–560, 2005.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Laurent Orseau and Mark Ring. Space-time embedded intelligence. In *International Conference on Artificial General Intelligence*, pp. 209–218. Springer, 2012.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krahenbuhl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. In *NeurIPS*, 2019.

Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. *arXiv preprint arXiv:2107.03380*, 2021.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pp. 9870–9879. PMLR, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. 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*, pp. 761–768, 2011.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *ArXiv*, abs/2002.10957, 2020.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

A Appendix

A.1 Learn cross-domain transferable successor features

Assume a neural network with parameters θ_{sf} is used to approximate the SF representation denoted as $\psi^{\pi_0}(x, a; \theta_{sf})$, where (x, a) comes from an arbitrary environment domain. Our aim is to use loss Eq. 4 to find the optimal θ_{sf}. By adding the Lagrangian multipliers λ_u and λ_l, the Lagrangian dual of Eq. 4 is:

$$\min_{\theta_{sf}} \quad \mathcal{L}_{sf} + \lambda_u I(\psi^{\pi_0}, Y) - \lambda_l \sum_{i=1}^{M} I(\psi^{\pi_0,i}, x^i) \tag{13}$$

Directly optimizing the mutual information (MI) terms of Eq. 13 in high-dimensional space is challenging, we provide tractable solutions by approximating the upper bound and lower bound.

Upper bound of $I(\psi^{\pi_0}, Y)$ Following (Feng et al., 2019), the $I(\psi^{\pi_0}, Y)$ is upper bounded by replacing one of the marginal distributions with a variational posterio distribution. For any distribution $q(Y)$, we can have an upper bound of $I(\psi^{\pi_0}, Y)$:

$$\begin{aligned} I(\psi^{\pi_0}, Y) &= \mathbb{E}_{p(\psi^{\pi_0}, Y)}[log\ p(Y|\psi^{\pi_0}) - log\ p(Y)] \\ &= \mathbb{E}_{p(\psi^{\pi_0})}[D_{KL}(p)((Y|\psi^{\pi_0})||q(Y)) - D_{KL}(p(Y)||q(Y))] \\ &\leq \mathbb{E}_{p(\psi^{\pi_0})} D_{KL}(p(Y|\psi^{\pi_0})||q(Y)) \end{aligned} \tag{14}$$

, where $q(Y)$ can be empirically estimated in a non-parametric way, like using kernel density estimation from the dataset. Therefore, minimization of the KL-divergence term will result in minimizing the mutual information term, which can be formulated as an adversarial training objective, as derived in (Feng et al., 2019):

$$\min_{\theta_{sf}} \max_{\theta_u} \mathcal{L}_u = \mathbb{E}_{p_{\theta_{sf}}(\psi^{\pi_0}, Y)}[log\ q_{\theta_u}(Y|\psi^{\pi_0}) - log\ q(Y)] \tag{15}$$

, where $q_{\theta_u}(Y|\psi^{\pi_0})$ is a classifier that predicts the probability of the sample belonging to label Y, and $q(Y)$ is a constant that can be dropped during optimization.

Lower bound of $I(\psi^{\pi_0,i}, x^i)$ There are multiple ways ((Oord et al., 2018; Nowozin et al., 2016)) to approximate a lower bound of the mutual information term. We follow (Feng et al., 2019) that maximizes the MI term by maximizing the Jensen-Shannon divergence form of MI, $\hat{I}^{JSD}(\psi^{\pi_0,i}, x^i; \theta_l)$, where θ_l parameterizes the JSD. Readers can refer to (Nowozin et al., 2016; Feng et al., 2019) for more details.

A complete loss term of Eq. 13 can be rewritten as a minimax optimization objective:

$$\min_{\theta_{sf}, \theta_l} \max_{\theta_u} \mathcal{L}_{sf} + \lambda_u \mathcal{L}_u - \lambda_l \sum_{i=1}^{M} \hat{I}^{JSD}(\psi^{\pi_0,i}, x^i; \theta_l) \tag{16}$$

, which is now tractable.

A.2 DQN-embodied

By using the projected Bellman updates (eq. 7 and eq. 8) proposed here, we compute the target values as described in eq. 9 and eq. 10. Then, the full DQN-embodied algorithm alternates optimizing Q^{π} and V^{π}_{proj} is as below:

Algorithm 2: DQN-embodied

Given pre-trained embodied set Ω^e, feature projection operator $\Pi_{\Omega^e}(x, a)$
Initialize $Q^\pi(.;\theta_u)$, $V^\pi_{proj}(.;\mathbf{w}_u)$, and replay buffer $\mathcal{D}$
for *i=1:N* **do**
  // Replay buffer
  for *t=0:T* **do**
    ϵ greedy select action a_t based on $max_a Q^\pi(x_t, a_t; \theta_{u,i})$
    Execute action a_t in environment, observe x_{t+1}, r_t
    Append transition sample (x_t, a_t, r_t, x_{t+1}) in $\mathcal{D}$
    Randomly sample batch transitions $\mathcal{B} = \{(x, a, r, x')\}$ from $\mathcal{D}$
    // Learn Q^π
    Set $y_i = \mathbb{E}_{(x,a,x')\sim\mathcal{T}^u}[r^u + \gamma V^\pi_{proj}(\Pi_{\Omega^e}(x', \arg\max_{a'} Q(x', a'; \theta_{u,i-1})); \mathbf{w}_{u,i-1})]$
    Perform gradient descent step on $\mathcal{L}_i(\theta_{u,i}) = (y_i - Q^\pi(x, a; \theta_{u,i}))^2$
    // Learn V^π_{proj}
    Set $y_{proj,i} = \mathbb{E}_{(x,a,x')\sim\mathcal{T}^u}[r^u + \gamma \max_{a'} Q^\pi(x', a'; \theta_{u,i-1})]$
    Perform gradient descent step on $\mathcal{L}_i(\mathbf{w}_{u,i}) = (y_{proj,i} - V^\pi_{proj}(\Pi_{\Omega^e}(x, a)); \mathbf{w}_{u,i})^2$
  end
end