DISENTANGLED CUMULANTS HELP SUCCESSOR REPRESENTATIONS TRANSFER TO NEW TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Biological intelligence can learn to solve many diverse tasks in a data efficient manner by re-using basic knowledge and skills from one task to another. Furthermore, many of such skills are acquired through something called latent learning, where no explicit supervision for skill acquisition is provided. This is in contrast to the state-of-the-art reinforcement learning agents, which typically start learning each new task from scratch and struggle with knowledge transfer. In this paper we propose a principled way to learn and recombine a basis set of policies, which comes with certain guarantees on the coverage of the final task space. In particular, we construct a learning pipeline where an agent invests time to learn to perform intrinsically generated, goal-based tasks, and subsequently leverages this experience to quickly achieve a high level of performance on externally specified, often significantly more complex tasks through generalised policy improvement. We demonstrate both theoretically and empirically that such goal-based intrinsic tasks produce more transferable policies when the goals are specified in a space that exhibits a form of disentanglement.

1 INTRODUCTION

Natural intelligence is able to solve many diverse tasks by transferring knowledge and skills from one task to another. For example, by knowing how to move objects in 3D space, it is possible to learn how to stack them faster. However, many of the current state-of-the-art artificial reinforcement learning (RL) agents struggle to do so. They are able to solve single tasks well, often beyond the ability of any natural intelligence (Silver et al., 2016; Mnih et al., 2015; Jaderberg et al., 2017), however even small deviations from the task that the agent was trained on can result in catastrophic failures (Lake et al., 2016; Rusu et al., 2016). Typically deep RL agents start learning every task from scratch. This means that each time they have to re-learn how to perceive the world (the mapping from a high-dimensional observation to state), and also how to act (the mapping from state to action), with the majority of time arguably spent on the former. The optimisation procedure naturally discards information that is irrelevant to the task, which means that the learnt state representation is often unsuitable for new tasks. Biological intelligence appears to operate differently. A lot of knowledge tends to be discovered and learnt without explicit supervision, through a process of *latent learning*, first suggested by Tolman (1948). This basic knowledge can then form the behavioural basis that can be used to solve new tasks faster. In this paper we argue that such transferable knowledge and skills should be acquired in artificial agents too (Barreto et al., 2018; Wulfmeier et al., 2019). In particular, we want to start by replicating the most basic form of latent learning found in biological intelligence – the ability to discover stable entities that make up the world and to learn basic skills to manipulate these entities. Compositional re-use of such skills enables biological intelligence to find reasonable solutions to many naturally occurring tasks, from goal-directed movement (controlling your own position), to food gathering (controlling the position of fruit and and nuts), or building a simple defence system (re-positioning multiple stones into a fence or digging a ditch). Similarly, artificial systems have been demonstrated to benefit from solving multiple tasks jointly, both in reinforcement learning (Torrey & Shavlik, 2010; Taylor & Stone, 2009) as well as supervised learning (Thrun & Pratt, 2012; Caruana, 1997).

To this end, we propose a principled way to learn a basis set of policies which can help agents quickly produce reasonable performance on the largest possible set of basic natural tasks within an environment. We restrict the set of tasks to those that can be expressed in natural language, and which do not rely on a particular execution ordering. We propose a method on how to discover these

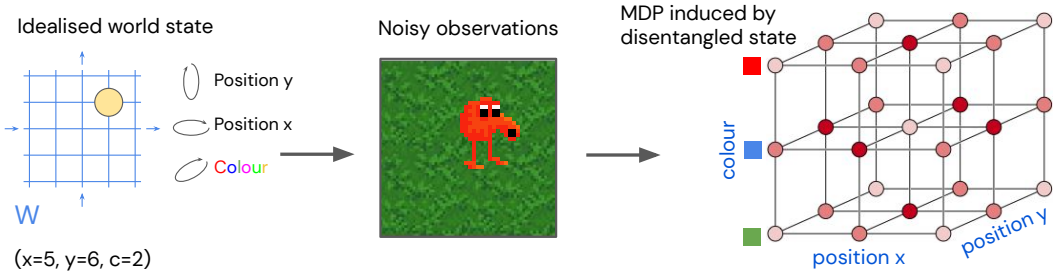

Figure 1: The idealised world state completely described by compositions of the following independent transformations: changes in position x, y and colour. Such a state may be projected into a high-dimensional observation, which may contain a lot of irrelevant detail, like the particular instantiation of the Qbert, or the grassy background. Disentangled representations recover the meaningful information about the independently transformable aspects of the world and disregard the irrelevant details.

policies in the absence of external supervision, a setting we call *endogenous reinforcement learning* (ERL), where the agent accumulates a transferable set of basic skills purely by intrinsic (endogenous) rewards which later builds the foundation to solve extrinsically (exogenously) provided tasks. In particular, we construct a learning pipeline where an agent invests time to learn to perform intrinsic, goal-based tasks in the ERL stage, and subsequently leverages this experience to perform few-shot learning of unseen tasks in the RL stage with certain guarantees on final task coverage. For this latter phase of few-shot learning, we make use of Generalized Policy Improvement (GPI) (Barreto et al., 2018) to leverage the behaviour learned in the intrinsic tasks. Moreover, we argue theoretically and empirically that such goal-based intrinsic tasks are more transferable to novel situations when the goals are specified via a representation that exhibits *disentanglement*.

Intuitively, disentangled representations consist of the smallest set of features that represent those aspects of the world state that are independently affected by natural transformations and together explain the most of the variance observed in an environment (Higgins et al., 2018) (see Fig. 1). Disentangled features, therefore, "carve the world at its joints" and provide a parsimonious representation of the world state that also points to which aspects of the world are stable, and which can in principle be transformed independently of each other. We conjecture that disentangled features align well with the idealised state space in which natural tasks are defined. Hence, by learning a set of policies that can control these features an agent will acquire a basis set of policies which spans a large set of natural tasks defined in such an environment. Note that both disentangled features and their respective control policies can be learnt without an externally specified task, purely in the ERL setting. We provide both theoretical justification for this setup, as well as experimental illustrations of the benefit of disentangled representations in a large set of tasks of varying difficulty.

Related work A number of past approaches share our motivation of discovering a diverse and useful set of policies in the absence of externally specified tasks. The predominant approach is to optimise an objective that encourages behaviours that are both diverse and distinguishable from each other (Gregor et al., 2017; Eysenbach et al., 2019; Hansen et al., 2019). Several other approaches investigated the idea of learning to solve intrinsic tasks specified in a learnt representation space (Nair et al., 2018; Laversanne-Finot et al., 2018). Some of these approaches have been demonstrated to be useful for solving downstream tasks, which could be either inferred (Eysenbach et al., 2019; Hansen et al., 2019) or specified (Nair et al., 2018). However, no theoretical guarantees have been provided concerning the downstream task coverage by the learnt set of policies. On the other hand, van Niekerk et al. (2018) and Barreto et al. (2017; 2018) provide theoretical guarantees on how well a given set of policies can be transferred to solve a wide range of downstream tasks. These papers, however, leave the question of how to discover and learn these policies open. Hence, our work provides a unique perspective by addressing both the questions of what makes a good basis set of policies to get certain guarantees on final task coverage, and how these policies may be learnt in the ERL setting. In general, shared training of systems to jointly solve a range of tasks as well as transfer and adaptation of previously trained models has provided strong performance gains across various domains. Work in supervised learning has build on parallel multitask (Caruana, 1997; Thrun & Pratt, 2012) as well as sequential transfer of models

trained on related tasks and data (Yosinski et al., 2014; Girshick et al., 2014). Similarly in reinforcement learning, transfer across related tasks has been proven efficient in accelerating training and simplifying exploration. Auxiliary tasks have been used to provide additional training signal (Jaderberg et al., 2017), share transition data (Riedmiller et al., 2018), or to generate transferable policies (Barreto et al., 2018; Wulfmeier et al., 2019). In particular, this work lies in the context of the latter but uses autonomously discovered basis tasks rendering it more scalable than manually chosen task distributions.

2 BACKGROUND

Basic Reinforcement Learning (RL) formalism. An RL agent interacts with its environment through a sequence of actions in such a way as to maximise the expected cumulative discounted rewards (Sutton & Barto, 1998). The RL problem is typically expressed using the formalism of Markov Decision Processes (MDPs) (Puterman, 1994). An MDP is a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of states and actions, $\mathcal{P}$ is the transition probability that predicts the distribution over next states given the current state and action $s' \sim \mathcal{P}(\cdot|s,a)$, $\mathcal{R}$ is the distribution of rewards $r \sim \mathcal{R}(s,a,s')$ received for making the transition $s \overset{a}{\mapsto} s'$, and $\gamma \in [0,1)$ is the discount factor used to make future rewards progressively less valuable. Given an MDP, the goal of the agent is to maximise the expected return $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$. This is done by learning a policy $\pi(a|s)$ that selects the optimal action $a \in \mathcal{A}$ in each state $s \in \mathcal{S}$. A typical RL problem attempts to find the optimal policy $\pi^* = \underset{\pi}{\text{argmax}}\, \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r | \pi\right]$, where the expectation is taken over all possible interaction sequences of the agent's policy with the environment. The optimal policy is learnt with respect to a particular task operationalised through the choice of the reward function $\mathcal{R}(s,a,s')$.

GPI & GPE Generalised Policy Improvement (GPI) and Generalised Policy Evaluation (GPE) (Barreto et al., 2017) can be used to transfer a set of existing policies to solve new tasks. The framework is specified for a set of MDPs:

$$\mathcal{M}^{\phi}(\mathcal{S},\mathcal{A},\mathcal{P},\gamma) = \{M^{\phi}(\mathcal{S},\mathcal{A},\mathcal{P},r,\gamma) \mid r(s,a,s') = \phi(s,a,s')^{\top} \boldsymbol{w}\} \tag{1}$$

induced by all possible choices of weights $\boldsymbol{w}$ that specify all possible rewards r, given a state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $\mathcal{P}$, discount factor γ and features $\phi(s,a,s')$. Note that the features are meant to be the same for all MDPs $M \in \mathcal{M}^{\phi}$. Given a policy π_i learnt to solve task i specified by $\boldsymbol{w}_i$, we can evaluate its value under a different reward $r_j = \phi(s,a,s')^{\perp} \boldsymbol{w}_j$ using GPE:

$$Q_j^{\pi_i}(s,a) = \mathbb{E}^{\pi_i}\left[\sum_{k=0}^{\infty} \gamma^k \phi_{t+k+1}(s,a,s') | \mathcal{S}_t = s, \mathcal{A}_t = a\right]^{\top} \boldsymbol{w}_j = \psi(s,a)^{\pi_i \top} \boldsymbol{w}_j. \tag{2}$$

Hence, given a set of policies π_1, π_2, ..., π_i induced by rewards r_1, r_2, ..., r_i over a subset of the MDPs $M' \subset \mathcal{M}^{\phi}$, we can get a new policy π_j for a new task induced by r_j (note that $M_j \in \mathcal{M}^{\phi}$, $M_j \cap M' = \emptyset$) according to:

$$\pi_j(s) = \underset{a}{\text{argmax}} \max_i \psi(s,a)^{\pi_i \top} \boldsymbol{w}_j. \tag{3}$$

3 ENDOGENOUS RL WITH GPE AND GPI

To illustrate the value of endogenous RL we propose the following three-part pipeline consisting of (1) a representation learning phase, (2) an intrinsic reinforcement learning phase and (3) a few-shot learning phase when a new task is presented, where steps 1-2 form the ERL stage, and step 3 forms the RL stage (see Fig. 2). This section provides a general overview of the pipeline. In Sec. 4 we will discuss why a particular version of the pipeline that uses features with the *disentanglement* property is expected to perform well in the RL stage.

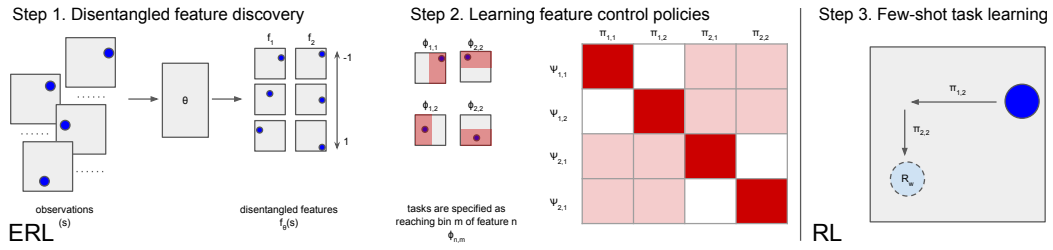

Figure 2: Schematic illustration of the three steps of our method. First we use an existing method for unsupervised disentangled feature discovery from observations obtained using an exploration policy. We then learn control policies that learn to achieve certain uniformly spread values for the learnt features. Finally, we use the feature control policies to solve tasks using the GPI framework. The first two steps do not require any extrinsic rewards.

Representation learning. At the beginning of the pipeline our agents learn a parameterized representation of the environment's state: $\phi_{1:n}(s) \in \Phi \subseteq \mathbb{R}^n$. Each feature ϕ_i will be used as a cumulant that gives rise to an RL task. Specifically, each feature ϕ_i will induce an associated optimal policy π_i that maximizes the expected discounted sum of ϕ_i.

As an illustration, we now describe a concrete way in which the features ϕ_i can be defined. Let $f_i : \mathcal{S} \mapsto [0,1]$ be arbitrary continuous functions, with $i = 1,2,...,k$. We define a discretization of each $f_i(s)$ into m uniformly spaced bins: $[0,1/m),[1/m,2/m),...,[(m-1)/m,1)$, denoted $b_1,...,b_m$ respectively. Using this notation, we can define a set of $n = mk$ features as follows:

$$\phi_{ij}(s,a) = \mathbf{1}\{f_i(s) \in b_j\}, \tag{4}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function. Note that, to facilitate the exposition, we use two indices to refer to a specific feature; obviously, we can "flatten" these indices and treat the features as a vector if needed. Intuitively, we can think of the task induced by $\phi_{ij}(s,a)$ as setting the i-th feature to a value in the interval b_j.

Intrinsic RL As discussed above, each feature ϕ_{ij} gives rise to an RL task. The second stage of our pipeline consists of solving these tasks. Crucially, instead of computing the value function of policy π_{ij} with respect to cumulant ϕ_{ij} only, we will compute the successor features of π_{ij} with respect to *all* cumulants:

$$\psi_{lh}^{\pi_{ij}}(s,a) = \mathbb{E}_{\pi_{ij}}\left[\sum_{t=0}^{\infty}\gamma^t \phi_{lh}(s_t,a_t) | s_0 = s, a_0 = a\right] \tag{5}$$

where $\pi_{ij}(s)$ is one of the optimal policies induced by cumulant ϕ_{ij}, that is, $\pi_{ij}(s) \in \text{argmax}_\pi Q_{ij}^{\pi}(s,\pi(s))$.

Collectively the successor features $\psi_{lh}^{\pi_{ij}}(s,a)$ can be thought of as an $n \times n$ matrix $\mathbf{\Psi}$ with cumulants in one dimension and policies in the other dimension. The matrix $\mathbf{\Psi}$ represents the agent's knowledge of how to manipulate the features of the environment. We will also use our double-subscript notation to refer to specific elements of $\mathbf{\Psi}(s,a)$: $\mathbf{\Psi}_{(lh),(ij)}(s,a) = \psi_{lh}^{\pi_{ij}}(s,a)$.

Few-shot learning phase Provided our agent has invested an initial effort to accurately learn the matrix $\mathbf{\Psi}$, we can now leverage it to perform few-shot learning. First, note that any linear combination of cumulants $\phi_{1:n}$, $\phi_w = \sum_i w_i \phi_i$, is itself a cumulant. We can then define the set of cumulants

$$\Phi = \left\{ \phi_w(s,a) = \sum_{i,j} w_{ij}\phi_{ij}(s,a) \,|\, w \in \mathbb{R}^{k \times m} \right\}. \tag{6}$$

Given an arbitrary task, we can find a cumulant $\phi_w \in \Phi$ that approximates the task as well as possible. One way to do so is to select $w \in \mathbb{R}^{k \times m}$ such that

$$w = \underset{w' \in \mathbb{R}^{k \times m}}{\text{argmin}}\, \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\left|\left| \sum_{ij} w'_{ij}\phi_{ij}(s,a) - r(s,a) \right|\right| \right], \tag{7}$$

where $\mathcal{D}$ is a distribution over $\mathcal{S}\times\mathcal{A}$ and $||\cdot||$ is a norm. Note that (7) is a linear regression.

Once we have computed $w\in\mathbb{R}^{k\times m}$, we can use the successor features of policy π_{ij} to evaluate it on task ϕ_w (a process sometimes referred to as Generalized Policy Evaluation or GPE):

$$Q_w^{\pi_{ij}}(s,a)=\sum_{l,h}w_{lh}\psi_{lh}^{\pi_{ij}}(s,a), \tag{8}$$

where $Q_w^{\pi_{ij}}(s,a)$ is the action-value function of π_{ij} under cumulant ϕ_w. Finally, by using GPI we can directly compute a policy based on the known policies π_{ij}:

$$\pi_w^{\text{GPI}}(s)=\operatorname*{argmax}_a\max_{ij}Q_w^{\pi_{ij}}(s,a). \tag{9}$$

While ϕ_w is not guaranteed to be optimal with respect to ϕ_w, its performance on this task is at least as good, and generally better, than that of the known policies π_{ij} (Barreto et al., 2017). Moreover, the computation of π_w^{GPI} is immediate given the matrix $\mathbf{\Psi}$ and w. Since we assume the matrix $\mathbf{\Psi}$ has been pre-computed in the ERL phase, this essentially reduces an RL problem to the regression problem shown in (7).

4 Theoretical Results

In this section we highlight the importance of the *choice* of latent representation used in our learning pipeline. Particularly, we show that when the agent's latent representation exhibits a particular form of disentanglement we can exploit this property to both accelerate the learning of our successor feature matrix and guarantee that GPI finds solutions to certain families of tasks.

Disentangled representations have recently been connected to symmetry transformations (Higgins et al., 2018), a powerful idea borrowed from physics. Roughly speaking, a symmetry transformation for a system is a transformation that leaves some property of the system unchanged. Here we use a specific definition of disentangled representation based on features that can be optimally controlled without affecting other features. More formally, we propose the following definition:

Optimal independent controllability Let $\phi_{1:n}$ be a set of features and let π_i^* be the optimal policy associated with the control task induced by the cumulant $\mathbf{1}\{\phi_i(s)\in\mathbb{R}_i\}$, with $\mathbb{R}_i\subset\mathbb{R}$. Let $(s_t)_{t=0}^N$ be a trajectory generated by following π_i^*. We call $\phi_{1:n}$ *optimally independently controllable* (OIC) if $\phi_j(s_t)=\phi_j(s_0)$ for all $j\neq i$ and $t\in\{1,N\}$.

Note that if we use the features defined in (4) we can have control tasks induced by $\mathbf{1}\{\phi_{ij}(s)=1\}=\mathbf{1}\{f_i(s)\in b_j\}$ for $j=1,2,...,m$. In this case two features ϕ_{ij} and ϕ_{hl} can be OIC only if $i\neq h$. We will abuse the terminology slightly and say that $f_{1:n}$ are OIC if any pair of the induced features ϕ_{ij},ϕ_{hl} is OIC when $i\neq h$. Without loss of generality we will assume henceforth that we are using features defined as in (4).

An immediate consequence of a set of OIC features is that values under rewards and policies associated with different features have a simple form:

Lemma 4.1. *When $f_{1:n}$ are optimally independently obtainable the successor feature matrix, $\mathbf{\Psi}$, has the following form:*

$$\mathbf{\Psi}_{(lh),(ij)}(s,a)=\begin{cases}\frac{1}{1-\gamma}\phi_{lh}(s,a) & \text{if } i\neq l\\ \Psi_{lh}^{\pi_{ij}}(s,a) & \text{else.}\end{cases} \tag{10}$$

Proof. The proof follows directly from the definition of OIC features. If $l\neq i$ then under $\pi_{i,j}^*$ $f_l(s_t)=f_l(s_0)$, and thus $\phi_{lh}(s_t,a_t)=\phi_{lh}(s_0,a_t)$, giving:

$$\begin{aligned}\Psi_{lh}^{\pi_{ij}^*}(s,a)&=\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\phi_{lh}(s_t,a_t)\big|\pi_{ij}^*,s_0=s,a_0=a\right]=\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\phi_{lh}(s_0,a_t)\big|\pi_{ij}^*,s_0=s,a_0=a\right]\\&=\frac{1}{1-\gamma}\phi_{lh}(s,a).\end{aligned} \tag{11}$$

□

Intuitively, Lemma 4.1 states that if the feature f_i associated with policy π_{ij} is different from the corresponding feature f_l used to define ϕ_{lh} the associated successor feature $\boldsymbol{\Psi}_{(lh),(ij)}(s,a) = \psi_{lh}^{\pi_{ij}}(s,a)$ reduces to $(1-\gamma)^{-1}\phi_{lh}(s,a)$. This reduces learning the matrix $\boldsymbol{\Psi}$ to learning a subset of its entries.

Guarantees for conjunctions of goal-based tasks In addition to simplifying the process of learning the successor feature matrix $\boldsymbol{\Psi}$, features with the OIC property come with guarantees under GPI for certain goal-based tasks. We define a goal-based task as one whose reward function has the form

$$R_G(s) = \mathbf{1}\{s \in G\} \tag{12}$$

where $G \subset \mathcal{S}$. Given the above definition, we say that a policy π "achieves" G if $V_{R_G}^{\pi}(s) > 0$ for all $s \in \mathcal{S}$.

Our uniform discretization of features $f_{1:k}$ into bins $b_{1:m}$ naturally induces a partition over of the state-space:

$$\mathcal{B}(\mathcal{S}) = \{B_{i_1,\dots,i_k} : i_1,\dots i_k \in [m]\} \tag{13}$$

where

$$B_{i_1,\dots,i_k} = \bigcap_{j=1}^{k} f_j^{-1}(b_{i_j}) \subset \mathcal{S}. \tag{14}$$

Intuitively, we can think of each partition $B_{i_1,\dots,i_k}$ as one of the possible m^k configurations of the features ϕ_{ij} (note that there are fewer than 2^{mk} configurations because some of them are impossible, as two bins associated with the same feature cannot be active at the same time). We can then think of these partitions as goal regions analogous to (12). We now show that for *any* goal $g \in \mathcal{B}(\mathcal{S})$ there exist a linear combination of the cumulants ϕ_{ij} that leads to a GPI policy that achieves g.

Theorem 4.1. *If $f_{1:k}$ are OIC, then for any $g \in \mathcal{B}(\mathcal{S})$ there exists a $w \in \mathbb{R}^{k \times m}$ such that π_w^{GPI} as defined in (8) and (9) achieves g. One such w is given by $w_{ij}^g = \mathbf{1}\{f_i(g) = b_j\}$.*

Proof. Recall that $\pi_{w^g}^{\text{GPI}} = \text{argmax}_a \; Q_{w^g}^{\max}(s,a)$, where $Q_{w_g}^{\max}(s,a) = \max_{ij} \sum_{lh} w_{lh}^g \Psi_{lh}^{\pi_{ij}^*}(s,a)$. We begin by rearranging terms in $Q_{w^g}^{\max}$:

$$\begin{aligned}
Q_{w^g}^{\max}(s,a) &= \max_{ij} \sum_{lh} w_{lh}^g \Psi_{lh}^{\pi_{ij}^*}(s,a) \\
&= \max_{ij} \sum_{h=1}^{m} w_{ih}^g \Psi_{ih}^{\pi_{ij}^*}(s,a) + \sum_{h=1}^{m} \sum_{l \neq i} w_{lh}^g \Psi_{lh}^{\pi_{ij}^*}(s,a) \\
&= \max_{ij} \sum_{h=1}^{m} w_{ih}^g \Psi_{ih}^{\pi_{ij}^*}(s,a) + \frac{1}{1-\gamma} \sum_{h=1}^{m} \sum_{l \neq i} w_{lh}^g \phi_{lh}(s,a) \\
&= \max_{ij} \sum_{h=1}^{m} w_{ih}^g \Psi_{ih}^{\pi_{ij}^*}(s,a) + \frac{1}{1-\gamma} \sum_{h=1}^{m} \sum_{l=1}^{k} w_{lh}^g \phi_{lh}(s,a) - \frac{1}{1-\gamma} w_{ih} \phi_{ih}(s,a) \\
&= C(s) + \max_{ij} \sum_{h=1}^{m} w_{ih}^g \left[\Psi_{ih}^{\pi_{ij}^*}(s,a) - \frac{1}{1-\gamma} \phi_{ih}(s,a) \right]
\end{aligned} \tag{15}$$

where the third equality follows from Lemma 4.1 and $C(s)$ captures $\phi_{lh}(s,a)$ terms (which do not depend on a or i).

First note that, from the form of w^g, for each i there is exactly one j such that $w_{ij}^g = 1$ with all other entries being 0. Denote this j as $b(i)$. We can then rewrite:

$$\begin{aligned}
Q_{w^g}^{\max}(s,a) &= C(s) + \max_{ij} \left[\Psi_{ib(i)}^{\pi_{ij}^*}(s,a) - \frac{1}{1-\gamma} \phi_{ib(i)}(s,a) \right] \\
&= C(s) + \max_{i} \left[\Psi_{ib(i)}^{\pi_{ib(i)}^*}(s,a) - \frac{1}{1-\gamma} \phi_{ib(i)}(s,a) \right]
\end{aligned} \tag{16}$$

Next notice that $\Psi_{ib(i)}^{\pi^*_{ib(i)}}(s,a) - \frac{1}{1-\gamma}\phi_{ib(i)}(s,a)$ is 0 if $f_i(s) \in b_{b(i)}$ and $\Psi_{ib(i)}^{\pi^*_{ib(i)}}(s,a)$ otherwise, giving:

$$Q_{w^g}^{\max}(s,a) = C(s) + \max_{i \in \mathbb{W}(s)} \Psi_{ib(i)}^{\pi^*_{ib(i)}}(s,a) \tag{17}$$

where $\mathbb{W}(s) = \{i : f_i(s) \notin b_{b(i)}\}$ or, more plainly, the set of feature indices that have not been achieved yet. This gives

$$\pi_{w^g}^{GPI}(s) = \operatorname*{argmax}_a Q_{w^g}^{\max}(s,a) = \operatorname*{argmax}_a \max_{i \in \mathbb{W}(s)} \Psi_{ib(i)}^{\pi^*_{ib(i)}}(s,a), \tag{18}$$

implying that $\pi_{w^g}^{\text{GPI}}$ will persistently pursue the "unachieved" feature ($\phi_{lh} = 0$) that is easiest to "achieve" (that is, to be set to $\phi_{lh} = 1$) among the features associated with nonzero elements in w^g. This means that eventually all such features will be set to 1, which in turn implies that goal g has been achieved. □

5 SPRITEWORLD EXPERIMENTS

In this section we experimentally validate that an agent can effectively use task-free interactions with an environment to gain a boost data efficiency across a wide range of subsequent tasks. In particular, we test whether an agent that uses GPI to transfer a set of feature control policies discovered in the ERL setting has a boost in performance over a baseline DQN agent that learns each task from scratch. We also validate whether disentangled feature control policies form a better basis for transfer compared to entangled feature control policies, and whether our pipeline outperforms DIAYN (Eysenbach et al., 2019), a state of the art approach for discovering a diverse set of policies in the absence of external tasks.

We validate our ideas on a toy Spriteworld environment (Watters et al., 2019) (see Fig. 3). The environment contains an agent and two sprites. The action space is 8-dimensional and consists of moving the agent up/down or left/right, as well as the same four actions but for dragging objects. It is only possible to drag objects if the agent is standing on them. Furthermore, dragging actions move the agent slower than the standard move actions. This environment makes it easy to define a wide range of diverse natural tasks of different difficulty level that can be easily expressed through language. The easiest tasks are specified in terms of the final position of the agent ("top", "bottom", "left" or "right"). The next difficulty level makes the locations more specific (e.g. "top left corner" or "bottom right corner"). We also specify equivalent tasks but in terms of the final object position (e.g. "square at the bottom", "circle at the top right corner"). Note that the object-based tasks are more difficult than the agent-based tasks, because agent's position can be controlled directly and independently in the action space, while controlling object position requires more elaborate policies that are also dependent on the agent's position. Finally, we also specify a set of tasks in terms of disjunctions of single agent or object properties (e.g. "agent to the left and square to the top"), conjunctions of object properties (e.g. "square at bottom right and circle at top left") or a combination of conjunctions and disjunctions (e.g. "circle at bottom right or top left"). The agent receives a reward of 1 if the relevant aspects of the environment state are within d distance of their respective goal locations, otherwise the reward is 0. Each episode terminates immediately if the goal is achieved. The agent and the objects are initialised in random positions sampled uniformly within the environment at the start of each episode. We evaluate the performance of our agent and the baselines on 3 tasks sampled from each of the 7 difficulty levels. Given the structure of our tasks, the average reward corresponds to the average number of episodes on which the agent solves the task.

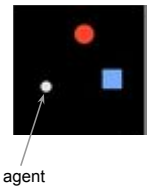

Figure 3: Spriteworld environment. The agent can move up/down, left/right and drag objects when it steps on them.

Ground truth features We evaluated how well our approach works in the scenario where disentangled features are the true x and y positions of the agent and the objects, and the entangled features are rotations of the disentangled features. Fig. 4 demonstrates that GPI over feature control policies provides an almost immediate boost in performance over the DQN baseline. This effect gets more prominent as the task difficulty increases, whereby the number of steps before DQN reaches the same performance as the GPI agent increases with task difficulty, whereby the GPI agent is able to solve the

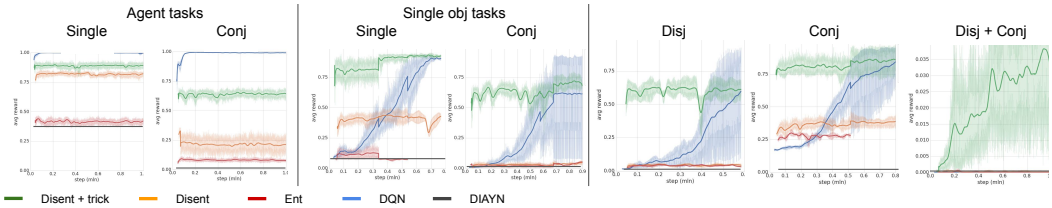

Figure 4: Average reward over all tasks for (left to right): 1) getting the agent to a location specified in terms of x or y; 2) getting the agent to a location specified in terms of a conjunction of x and y; 3-4) same for the agent; 5) getting two objects or an object and an agent to a disjunction of two locations specified in terms of an x or a y position; 6) same as before but now to a conjunction of locations specified in terms of an x or a y position; 7) same as before, but now the location is specified as a disjunction of two conjunctions. Note that GPI over disentangled features with the "off-diagonal" trick is able to solve all tasks (note that the average reward indicates the proportion of episodes on which tasks are solves), incluing the hardest disjunction of conjunctions tasks that even DQN could not solve.

tasks most of the time within around 50k learning steps, while the DQN baseline typically requires > 150k steps. We also see that GPI over disentangled features provides a significantly bigger jump in performance compared to GPI over entangled features. Finally, it is clear that the "off-diagonal" trick works well for the disentangled GPI, but not for the entangled version, which suggests an additional benefit of better computational efficiency during ERL learning for the former version of our pipeline. Moreover in our comparisons against DIAYN, we found that the competing method could only learned to perform tasks involving the agent, failing to learn to interact with other objects.

6 Conclusions

We have proposed a principled way to learn and recombine a basis set of policies that guarantees achievability for a large set of natural conjunctive tasks in an environment. We have demonstrated that these policies can be learnt in the ERL setting, where the agent has no access to external rewards and has to learn by setting its own tasks. We theoretically justified and experimentally verified the three-stage pipeline, where the agent discovers useful features, learns to achieve endogenously specified tasks of setting these features to particular values, and then used GPI over these policies to bootstrap reasonable performance on a wide range of natural tasks which can be expressed in language and which do not rely on a particular execution ordering. We demonstrated that disentangled features produce better task coverage and learning efficiency, since they reflect those aspects of the environment that can in principle be transformed independently of each other.

References

Andre Barreto, Will Dabney, Remi Munos, Jonathan Hunt, Tom Schaul, Hado van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *NIPS*, 2017.

Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. *NIPS*, 2018.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: learning skills without a reward function. *ICLR*, 2019.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

Karol Gregor, Danilo Rezende, and Daan Wierstra. Variational intrinsic control. *ICLR*, 2017.

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

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv*, 2018.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *ICLR*, 2017.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, pp. 1–101, 2016.

Adrien Laversanne-Finot, Alexandre Péré, and Pierre-Yves Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. *arxiv*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David S Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *arxiv*, 2018.

M. L. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. John Wiley Sons, Inc., 1994.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Tobias Springenberg. Learning by playing – solving sparse reward tasks from scratch. *ICML*, 2018.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arxiv*, 2016.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

Edward C. Tolman. Cognitive maps in rats and men. *The Psychological Review*, 55(4):189–208, 1948.

Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pp. 242–264. IGI Global, 2010.

Benjamin van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Composing value functions in reinforcement learning. *NeurIPS*, 2018.

Nicholas Watters, Loic Matthey, Sebastian Borgeaud, Rishabh Kabra, and Alexander Lerchner. Spriteworld: A flexible, configurable reinforcement learning environment. https://github.com/deepmind/spriteworld/, 2019. URL `https://github.com/deepmind/spriteworld/`.

Markus Wulfmeier, Abbas Abdolmaleki, Roland Hafner, Jost Tobias Springenberg, Michael Neunert, Tim Hertweck, Thomas Lampe, Noah Siegel, Nicolas Heess, and Martin Riedmiller. Regularized hierarchical policies for compositional transfer in robotics. *arXiv preprint arXiv:1906.11228*, 2019.

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

A APPENDIX

A.1 SPRITEWORLD ENVIRONMENT

The Spriteworld environment consists of a room without obstacles with an agent and two objects: a circle and a square. The agent can take 8 agents consisting of 4 regular movements (up, down, left, right) and 4 "dragging movements" which mirror the regular movements but move the agent half as far. Objects in the environment can overlap and pass through each other. When the agent overlaps with an object and executes a dragging movement, both the agent and the object are moved together. For our experiments we use a vectorized version of our environment state as observations to our models, consisting of 6 scalars representing the the x and y positions of the agent and each object. When the environment is reset, both the agent and objects are randomly positioned.

A.2 LEARNING THE SUCCESSOR FEATURE MATRIX

For all experiments involving learning the successor feature matrix $\mathbf{\Psi}$ we trained a parameterized Q network. This network takes a state and target policy as input and outputs a vector of successor features under the target policy. The entire vector of successor features under a policy are updated at once for each transition sampled from that policy. We generate behavior through a collection of 128 actors, each of which commits to pursuing a single cumulant ϕ_{ij} and follows an epsilon greedy policy with respect to that policy.

Our parameterized Q network consists of one fully connected layer of width 1024 applied to the observation and policy representations before combining taking their product to combine them. Following this product, we apply two more fully connected layers of size 1024 before outputting our final vector of successor features. All network activations are leaky relus with $\alpha = 0.1$.