# Learning to Predict Consequences as a Method of Knowledge Transfer in Reinforcement Learning

Eric Chalmers, Edgar Bermudez Contreras, Brandon Robertson, Artur Luczak, and Aaron Gruber

***Abstract*— The reinforcement learning (RL) paradigm allows agents to solve tasks through trial-and-error learning. To be capable of efficient, long-term learning, RL agents should be able to apply knowledge gained in the past to new tasks they may encounter in the future. The ability to predict actions' consequences may facilitate such knowledge transfer. We consider here domains where an RL agent has access to two kinds of information: agent-centric information with constant semantics across tasks, and environment-centric information, which is necessary to solve the task, but with semantics that differ between tasks. For example, in robot navigation, environment-centric information may include the robot's geographic location, while agent-centric information may include sensor readings of various nearby obstacles. We propose that these situations provide an opportunity for a very natural style of knowledge transfer, in which the agent learns to predict actions' environmental consequences using agent-centric information. These predictions contain important information about the affordances and dangers present in a novel environment, and can effectively transfer knowledge from agent-centric to environment-centric learning systems. Using several example problems including spatial navigation and network routing, we show that our knowledge transfer approach can allow faster and lower cost learning than existing alternatives.**

***Index Terms*— Egocentric, navigation, network routing, reinforcement learning (RL), transfer learning.**

## I. Introduction

ONE of the most important aspects of intelligent behavior is the ability to learn and adapt to changing situations [1]. This often requires agents to apply previously acquired knowledge to similar but novel situations, an ability known as *transfer learning*. Transfer learning has been studied extensively in psychology, and is a key concept related to learning and knowledge acquisition [2]–[6]. In more practical terms, transfer learning can be understood as the capability to "learn to learn" [7], so that as an agent learns, its ability to learn increases. In artificial intelligence, the development of systems capable of transfer learning remains a challenge.

Reinforcement learning (RL) is a machine learning paradigm in which an agent learns suitable behavior through interaction with its environment [8]. Traditional RL approaches model the environment as a Markov decision process (MDP) consisting of a set of discrete states $S = \{s_1, s_2, s_3 \ldots\}$, and actions $A = \{a_1, a_2, a_3 \ldots\}$ that may be executed from each state. Executing action $a$ from state $s$ causes a transition to state $s'$ according to a *transition function* describing $P(s'|s, a)$, and triggers a reward $r$ according to a *reward function*, $r = R(s, a, s')$. If the agent's objective is to maximize reward, the value $Q$ of executing action $a$ from state $s$ is

$$Q(s, a) = E[r + \gamma \times \max_{a'}(Q(s', a'))] \tag{1}$$

where $\gamma \epsilon$ [0,1] is a discount factor controlling the agent's degree of preference for immediate rewards over future ones. The agent typically has no prior knowledge of the transition and reward functions, and must learn value estimates iteratively as it explores the environment. Individual value estimates learned in one task may or may not be valid in another: transfer learning techniques identify conditions that permit transfer, and the specific pieces of knowledge that are transferrable.

In this paper, we focus on a particular kind of transfer that is possible when the agent has access to an agent-centric space of state variables with constant semantics across tasks, and an environment-centric space of state variables whose semantics may change across tasks.

As an example, imagine a search-and-rescue robot designed to detect injured people inside damaged buildings. The robot is equipped with cameras, microphones, and other sensors, as well as a dead-reckoning system that estimates the robot's location in the building. The sensor readings in this example form an agent-centric space with consistent semantics across tasks—a human call for help means the same thing in one building as it does in another. The location information, on the other hand, forms an environment-centric space with differing semantics across tasks: a location 10 m from the exit may conceal an injured person in one building, and a hazardous fire in another. In these types of problems, transfer learning can be achieved (and environment-centric learning can be accelerated) by transferring knowledge from agent-centric to environment-centric systems.

### A. Related Work

RL is an active research field, attractive in artificial intelligence and robotics [9] as a very natural trial-and-error style of learning. Transfer learning is a subject of considerable research effort in the RL field, due to its importance as a key part of

Manuscript received May 31, 2016; revised September 29, 2016 and January 19, 2017; accepted March 25, 2017. Date of publication April 17, 2017; date of current version May 15, 2018. This work was supported by the Natural Sciences and Engineering Research Council of Canada, under Grant 331011712. *(Corresponding Author: Eric Chalmers.)*

The authors are with the Department of Neuroscience, the University of Lethbridge, Lethbridge, AB T1K 3M4, Canada (e-mail: dchalmer@ualberta.ca; edgar.bermudez@uleth.ca; robertsonb@uleth.ca; luczak@uleth.ca; aaron.gruber@uleth.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2017.2690910

efficient long-term learning. We refer the reader to [10] for a thorough review of knowledge transfer methods in RL, and provide a brief overview of the relevant and recent methods here.

Some transfer learning methods employ hierarchical behavioral control, such as that proposed in [11]. Mulling *et al.* [12] present a framework in which a robot learns several motor primitives, and then transfers knowledge of the motor primitives to the larger task of playing table tennis. In [13], a robot learned low-level grasping motions, and then used RL at a higher level to combine these motions into a successful object-grasping program. Hart and Grupen [14] advocate an agent-centric rather than an environment-centric representation of behavior, which is highly relevant here. They show that by relying on an agent-centric representation, transferrable skills may be learned, from which hierarchical programs may be assembled.

Hierarchical abstraction can be applied in the state and action spaces as well. Chalmers *et al.* [15], [16] and Shoeleh *et al.* [17] created hierarchical abstractions of the state space, finding that knowledge gained at the abstract level can be transferred between tasks. Konidaris and Barto [18] showed that options (i.e., a particular sequence of actions) learned in an agent-centric space can be transferred between tasks with different environment-centric spaces. Fuzzy RL [19]–[22] can also be applied to such a hierarchical framework [23], [24].

One alternative to hierarchical learning is imitation learning [25], which can allow knowledge transfer from a teacher to a learning agent, though the agent is likely to discover only solutions that are similar to what they have observed during demonstration [9]. Conn and Peters [26] and Wang *et al.* [27] demonstrate how RL agents may learn from human demonstration. A related approach (which is relevant here) transfers knowledge to the agent in the form of an initial policy [e.g., an initial table of $Q(s, a)$ values to be used as a starting point in learning]. For example, [28] and [29] provided robots with a good generic policy, which the RL process then adapted and optimized for specific situations. Pastor *et al.* [30] derived this initial policy from human demonstration, but also endowed their robot with the ability to record sensor readings and predict future readings. These predictions were used as a form of failure detection.

The recent work in [31] goes beyond using predictions for failure detection. In their framework, the agent generates prototypes (hypothetical experiences) based on past experiences in source tasks, and uses these prototypes to accelerate learning in a new, target task. Their particular framework applies only when source and target tasks share a state space, but the core idea of learning from predictions is highly relevant to our work.

This paper focuses on transfer learning in RL, but we should note that transfer learning is an important theme throughout all branches of artificial intelligence. Chaturvedi *et al.* [32] implemented deep transfer learning in an artificial neural network in order to classify Gaussian networks with time-delays. They propose to learn to classify high-dimensional networks using previously trained low-dimensional subnetworks or "motifs." In their approach, Kullback–Leibler (KL) divergence [33] was used to minimize the distance between source and target motifs to transfer knowledge.

This paper considers RL knowledge transfer between tasks that may have different environment-centric state spaces, but share an agent-centric state space with semantics that remain constant across tasks. Many researchers have approached such situations by performing RL in the agent-centric and environment-centric spaces separately. Whenever the agent is in a novel environment-centric state, the agent-centric learning system can provide an initial policy—the rationale being that the agent-centric system has at least some knowledge of action values given the current agent-centric state. For example, our search-and-rescue robot has no initial environment-centric knowledge at its first exposure to an unfamiliar building: a situation that normally prompts a random exploratory action. However, if the robot has learned agent-centrically that the fire detected by its infrared cameras should be avoided, then this knowledge can be transferred to the environment-centric system as a low initial value for actions leading toward the fire or, equivalently [34], as a "shaping" function which biases the agent against moving toward fires.

This initial-policy approach has been implemented in a variety of ways. Konidaris *et al.* [35] recently developed a general framework that transferred policies from agent-centric to environment-centric systems in this way. Banerjee and Sone [36] used a similar approach to transfer knowledge between games. Guofang *et al.* showed that agents could effectively learn from humans in an agent-centric space, then transfer this knowledge to environment-centric space by $Q$ value initialization [37]. Mousavi *et al.* [38], [39] also initialized $Q$ values in target tasks to those from source tasks where MDP homomorphism makes an explicit mapping from source to target state—action pairs possible. An interesting variation by Freire and Costa [40] and Koga *et al.* [41], accelerates early learning by consulting action values for environment-centric states in source tasks that share the same agent-centric representation as the state in the target task. Maron *et al.* [42] explored the concept of "affordances," an analog of agent-centric features applied to object-oriented-MDPs, which accelerate environment-centric learning by pruning the action set based on the current agent-centric information.

The literature has demonstrated that *$Q$ value* initialization or "shaping" is effective in some types of tasks. However, *$Q$ value* initialization simply gives the agent a bias for or against executing certain actions from a given state; it tells the agent nothing about the consequences of those actions. As we will show in this paper, information about actions' consequences is important if the task requires learning a long sequence of actions, or if the agent-centric information does not clearly indicate an optimal action.

### *B. Predicting Consequences: New Opportunity for Knowledge Transfer*

Generating simple predictions about actions' consequences is critically important for biological agents. From low-level brain circuits that predict future sensory input [43], [44] to higher level learned associations between sensory input and predictable outcomes [45], the mammalian brain has an extraordinary ability to make predictions. Indeed, even simple actions like walking would be impossible without the brain's ability to predict body motion and position a short time into the future [46]. Several recent developments in RL seem to have been inspired by the importance of forecasting in biological learning, and have combined RL with real-time predictions for improved prosthetic control [47], novelty detection [48], and to prove that a simple robot can produce thousands of real-time predictions about its environment [49]. However, the utility of predictions for knowledge transfer is yet to be explored.

### C. Contribution of This Paper

Here, we propose an RL system in which an agent learns to use agent-centric information to predict actions' environmental consequences—in much the same way as we might predict where a doorway leads, or the consequences of drinking a beverage that appears to be very hot. The agent learns from these predictions as well as its real experiences in the environment. This ability to predict then becomes an effective way to transfer knowledge between any tasks that share a common agent-centric space. While prediction is a central theme in machine learning generally, to our knowledge, no other work has used predictions as a means of knowledge transfer in RL.

We test the proposed approach in three domains: grid-world-style navigation tasks, more complex spatial navigation tasks involving a simulated robot, and a simple networking problem unrelated to navigation. We show its applicability to both model-free and model-based learning. We compare with two other recent and representative methods designed for the same class of problems—involving both agent-centric and environment-centric information. The new prediction-based approach to knowledge transfer shows faster convergence and lower cost of learning in these tasks, even if the agent's predictions are occasionally erroneous. Our approach can be used as a general framework for tackling problems with agent-centric and environment-centric components.

## II. Algorithm

We consider an agent exposed to a set of environments modeled as MDPs. Borrowing some notation from [35], we write the $i$th MDP in the set as

$$M_i = < S_i, A_i, P_i, R_i, D > .$$

$S$ is the set of environmental states defined on environment-centric state variables. $D$ is the agent-centric space, which could be a set of distinct agent-centric states ($D = \{d_1, d_2, d_3 \ldots\}$) or a continuous space created by $n$ agent-centric features ($D \epsilon R^n$), but is defined consistently across all tasks. The agent's sensors map each state $s$ in each environment to an agent-centric state $d \epsilon D$. Knowledge transfer is possible between *related* MDPs $M_i$ and $M_j$ if they share a nonempty agent-centric space.

We now make two important assumptions

$$P_i(s'|a, s) = f(d, s, a)$$
$$r = g(d, a).$$

That is, individual transition probabilities and rewards from state $s$ can be estimated as a function of the corresponding agent-centric state. This requires that the environmental state space S have consistent dynamics (making state transitions at least partially predictable) and that the agent is equipped with sensors sufficient to allow the predictions; though we will show that our approach is advantageous even if transitions and rewards cannot always be predicted perfectly. Navigation-style problems are a good example of a domain fitting the above description. In a navigation problem, the environment-centric state $s$ may encode the agent's absolute location in the environment, while the agent-centric state $d$ encodes the location of roads, obstacles, and so on, relative to the agent. Given an environment-centric state $s$ (e.g., "10 m West, 5 m North, heading due East"), a sufficiently detailed agent-centric state $d$ (e.g., "obstacle to the right, all clear ahead"), and an action $a$ (e.g., "move forward 1 m"), the subsequent environment-centric state can usually be predicted ("9 m West, 5 m North, heading due East").

Kearns and Koller [50] discovered some time ago that, for the many real-world problems in which transition probabilities are predictable, this predictability can be exploited to significantly reduce the computational complexity of RL in a single environment. Here, we present a framework that exploits predictability in the environment-centric space to achieve knowledge transfer *between environments* that share a common agent-centric space. The framework consists of an environment-centric RL system, which learns to solve the task $M_i$ in the environment-centric space $S_i$. This RL system may be model-free or model-based an agent-centric learning system (not necessarily RL), which learns to predict $P(s'|a,s)$ and $r$ given $s$, $d$, and $a$.

The algorithm is illustrated in Fig. 1 and detailed in Algorithm 1. The agent observes the current environment-centric and agent-centric states, $s$ and $d$. For each action $a \epsilon$ A, the agent-centric learning system predicts the consequent state $s'$ and reward $r$ that will result if action $a$ is executed from state $s$. We denote these predictions as $\hat{s'}$ and $\hat{r}$. A predicted state transition $< s, a, \hat{s}', \hat{r} >$ is then formed and used to update the environment-centric RL system, as if the predicted transition had actually happened. However, the update is conditional on a predicate $\Psi$, which we call the *knowledge transfer predicate*.

$$\psi : (s, d, a) \longmapsto \{\text{true, false}\}.$$

The knowledge transfer predicate helps ensure that predicted transitions are used only when necessary, and when the predictions are of reasonable quality. For example, $\Psi$ might evaluate to *true* only when $< s, a >$ is a novel state–action pair and when $< d, a >$ has been experienced before, or it may take into account the confidence in the $\hat{s'}$ and $\hat{r}$ predictions; the exact implementation of $\Psi$ will likely depend on the nature of the agent-centric learning system.

After knowledge has been transferred in the form of predicted transitions, the environment-centric RL system selects an action $a'$ to perform. It is then updated using the resulting (real) transition $< s,a',s',r >$. A training observation for

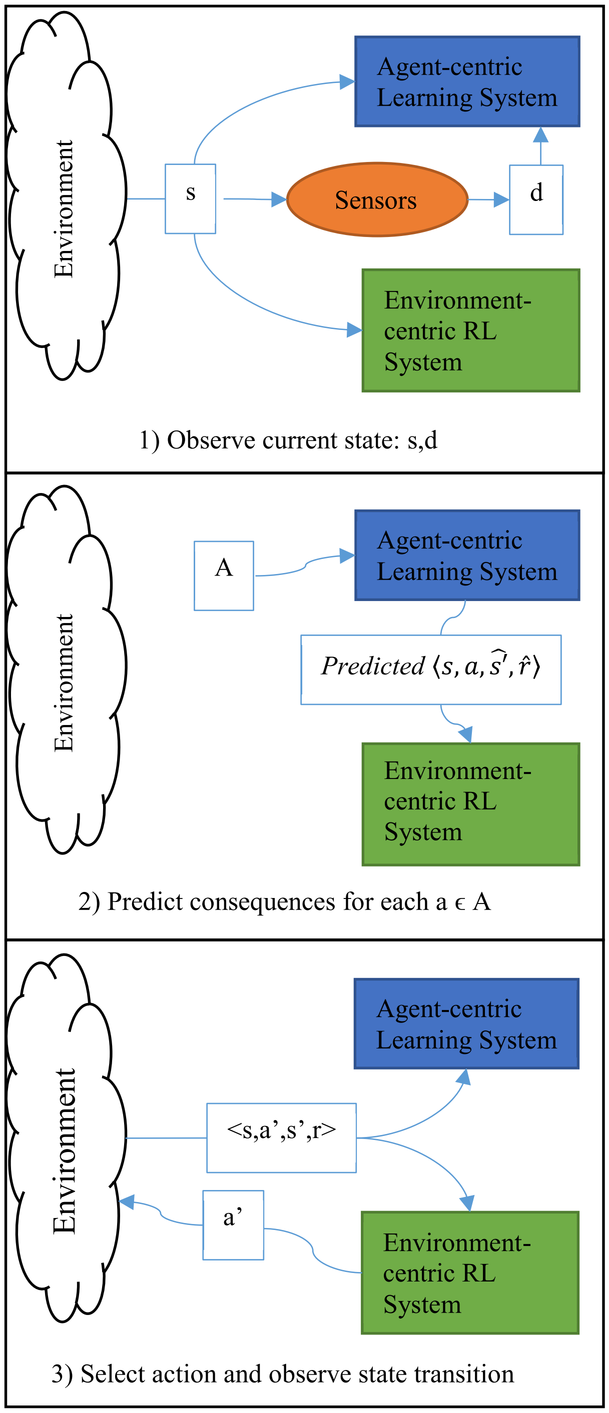
Environment
Agent-centric Learning System
s
Sensors
d
Environment-centric RL System
1) Observe current state: s,d
A
Predicted ⟨s, a, ŝ′, r̂⟩
2) Predict consequences for each a ϵ A
<s,a',s',r>
a'
3) Select action and observe state transition

Fig. 1. Illustration of the proposed knowledge transfer approach. 1) After observing the current agent-centric and environment-centric states, 2) the agent-centric learning system predicts the consequences of each available action. 3) The predicted transitions are used to update the environment-centric system as if they actually occurred, ultimately allowing it to make a more informed selection of an action to execute. The real transition resulting from this action is used to update both learning systems.

the agent-centric system is also created, with $s$, $d$, and $a'$ as inputs and $s'$ and $r$ as targets.

Algorithm 1 is a learning meta-algorithm [51], which relies on environment-centric and agent-centric components of a designer's choosing. The time complexity of the algorithm will be a function of the complexity of these components: the complexity of RL has been considered for both

**Algorithm 1** Pseudocode for Knowledge Transfer in RL by Predicting Actions' Consequences

```
Inputs
    A = {a_1, a_2, a_3 ...} : the set of allowed actions
Local
    L_a : agent-centric learner
    L_e : environment-centric reinforcement learner
    Ψ : knowledge transfer predicate
Loop:
    observe current states: $d, s$
    Foreach action a ϵ A:
    If $\Psi(s, d, a)$
        predict $\langle \hat{s}', \hat{r} \rangle = L_a(s, d, a)$
        update L_e with predicted transition $\langle s, a, \hat{s}', \hat{r} \rangle$

    select and execute action $a'$
    observe new state and reward: $s', r$
    update L_e with transition $\langle s, a, s', r \rangle$
    train L_a using inputs $\langle s, d, a' \rangle$ and outputs $\langle s', r \rangle$
```

model-free [52], [53] and model-based cases [54], [55], though the agent-centric learning system need not be RL-based at all. In general, we simply note that the environment-centric RL system is updated at least once and at most $1 + |A|$ times each step, the knowledge transfer predicate is evaluated $|A|$ times per step, the agent-centric system generates at most $|A|$ predictions each step, and the agent-centric system is trained or updated, at most, once per step.

## III. Testing

We tested the applicability of our approach in three example situations: two from the navigation domain and one simple network routing problem. We have made a deliberate effort to implement the proposed transfer learning scheme somewhat differently in each experiment, illustrating its versatility in accommodating a variety of agent-centric and environment-centric learning components, including model-free and model-based approaches.

In each test, we compare with two other knowledge transfer approaches from the literature. The first was proposed in [35] in 2012 and performs agent-centric and environment-centric RL separately. $Q$-values for novel environment-centric states are initialized by transferring the values from the agent-centric learner, given the associated agent-centric state. We will refer to this method as "transfer by shared features," in accordance with this paper's title.

The second comparison method was the best of a set of methods proposed in [40] in 2015. Given an environment-centric state with a corresponding agent-centric state, this method identifies a set of environment-centric states from previous tasks with the same agent-centric state. An "abstract policy" is derived by building the set of actions that were optimal in these other environment-centric states, and randomly selecting one for execution. The abstract policy is invoked at each step with a probability that decays linearly from 1 to 0 over the course of $\tau$ steps; the agent uses an $\varepsilon$-greedy environment-centric policy otherwise. In our experiments we

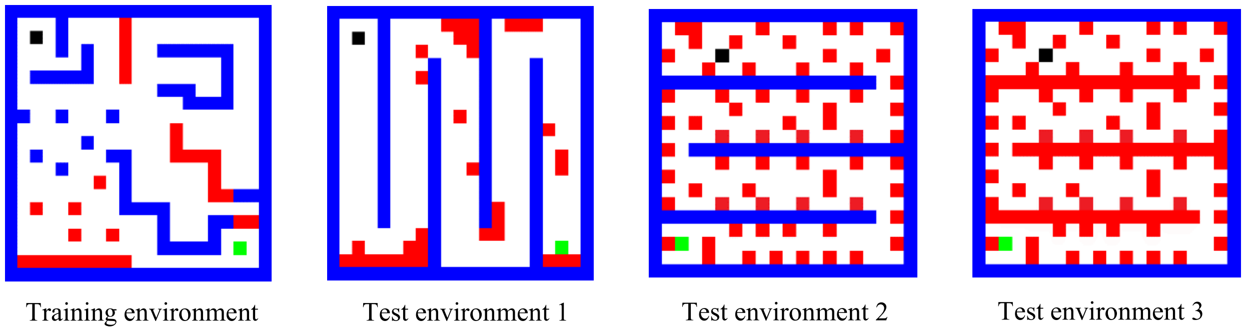

Fig. 2. Grid world environments used in the test. Blue patterns represent walls that the agent cannot traverse, green squares represent the goal state, and red patterns represent lava that can be traversed with a large negative reward. The agent learned in the training environment before being evaluated in the test environments, where the ability to transfer knowledge about affordances and risks of various world elements can accelerate learning.

used $\tau$ values of 100, 1000, 10 000, and 100 000, reporting the best result (In every case, we found that the value of $\tau$ had only a minor effect on performance). We will refer to this method as "transfer by abstract policy."

### A. Example 1: Grid World Navigation

We first tested our proposed method in grid world simulations (Fig. 2). The agent was allowed four actions corresponding to stepping up, down, left, and right by one square. The environment contained four distinct elements: open space, walls, lava, and a single goal. Navigating through open space triggered a reward of $-0.01$, attempting to navigate into a wall resulted in no movement and a reward of $-0.1$, navigating through lava triggered a reward of $-1$, and navigating into the goal triggered a reward of $+10$ and teleported the agent back to the start state.

At each step, the agent was informed of the $x$ and $y$ coordinates of its current location in the environment (the environment-centric state $s$) and was allowed to sense the four directly adjacent cells, including their element type (the agent-centric state $d$).

Agents were allowed to learn in a training environment containing the four environmental elements described previously for 400 000 steps before being evaluated in three novel test environments. Any environment-centric knowledge was cleared during the transition between environments (i.e., agents were informed of each task change). This test was intended to test agents' ability to agent-centrically learn the affordances and dangers of various elements, and then transfer this knowledge to accelerate learning in the new test environments.

*1) Algorithm Implementation:* Algorithm 1 was implemented as follows for the grid world environments. The environment-centric reinforcement learner was implemented as a Q learning algorithm [8]. The Q learning algorithm maintained a table Q[$s$,$a$] of Q values for each state–action pair. Actions were selected using an $\varepsilon$-greedy policy with $\varepsilon = 0.05$, meaning that the agent selected a random action with probability 0.05 and selected the action with the highest value otherwise. Given a state transition $< s,a,s',r >$, the Q value for the corresponding state–action pair was updated as

$$Q[s,a] = Q[s,a] + \alpha(r + \gamma \max_{a'} Q[s', a'] - Q[s,a]) \tag{2}$$

where the learning rate $\alpha$ was set to 0.2, and $\gamma$ was set to 0.9.

The agent-centric learning system was implemented as an array of three independent Q learning algorithms that learned to predict $\Delta x$ (the change in $x$ coordinate), $\Delta y$ (the change in $y$ coordinate), and $r$ resulting from taking action $a$ from agent-centric state $d$. This was accomplished by supplying the Q learning algorithms with $\Delta x$, $\Delta y$, and $r$ as reward signals, as in [49] and [56]. Learning rates $\alpha$ were set to 0.2. All three discount factors were set to $\gamma = 0$, so that the *immediate* $\Delta x$, $\Delta y$, and $r$ would be predicted; setting $\gamma > 0$ would cause the predictions to extend a number of steps into the future [49], a potentially interesting case considered in Section IV. The predicted transition $< s, a, \hat{s}', \hat{r} >$ was constructed based on the $\Delta x$, $\Delta y$, and $r$ predictions. Note that any supervised learning technique could be used in place of the three $Q$ learning algorithms.

Finally, tables $T_a[d,a]$ and $T_e[s,a]$ were used to encode the number of times each $< d,a >$ and $< s,a >$ state–action pair was encountered. The knowledge transfer predicate was implemented simply as

$$\psi(s,d,a) = (T_a[d,a] > t_a) \;\&\&\; (T_e[s,a] > t_e). \tag{3}$$

That is, predictions were only generated and used to transfer knowledge if the agent had experienced agent-centric state $d$ more than $t_a$ times, and experienced the environment-centric state $s$ more than $t_e$ times. The implementation with a model-based environment-centric learner used $t_a = t_e = 1$. The implementation with $Q$ learning as the environment-centric learner used $t_a = 1$ and $t_e = 1000$, since it was assumed that the slower $Q$-learning algorithm would benefit from repetitive knowledge transfer.

The transfer by shared features approach was implemented using $Q$-learning algorithms for both agent-centric and environment-centric components. The agent-centric $Q$-learning algorithm used $\alpha = 0.1$ and $\gamma = 0.9$, while the environment-centric algorithm used $\alpha = 0.2$ and $\gamma = 0.9$. The transfer by abstract policy approach was implemented using $Q$-learning with $\alpha = 0.2$ and $\gamma = 0.9$.

*2) Results:* Figs. 3–5 show cumulative reward obtained by agents in test environments 1, 2, and 3, respectively. In these plots, the minimum value indicates the cost of learning, the zero-crossing point indicates the time required to recoup the

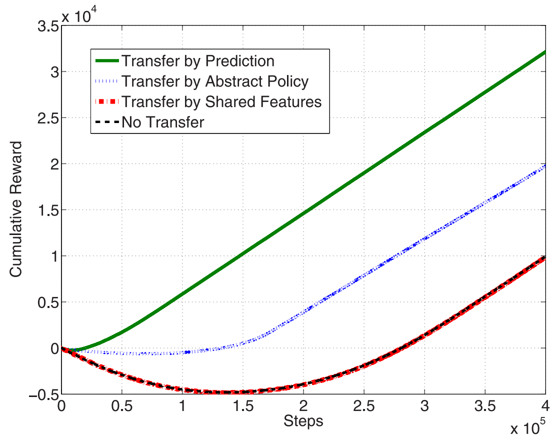

Fig. 3. Cumulative reward obtained by agents in the grid world test environment #1.

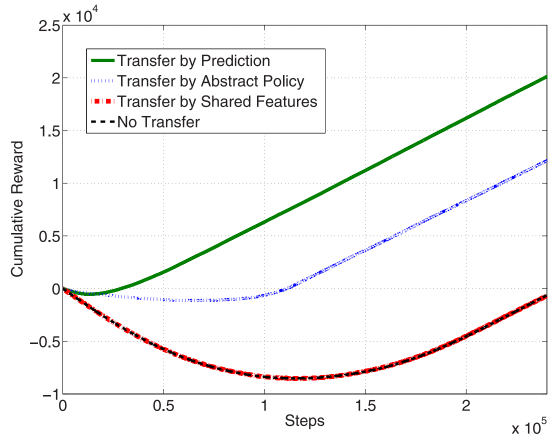

Fig. 4. Cumulative reward obtained by agents in the grid world test environment #2.

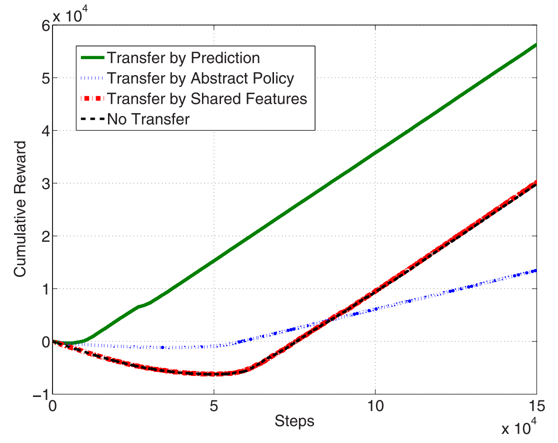

Fig. 5. Cumulative reward obtained by agents in the grid world test environment #3.

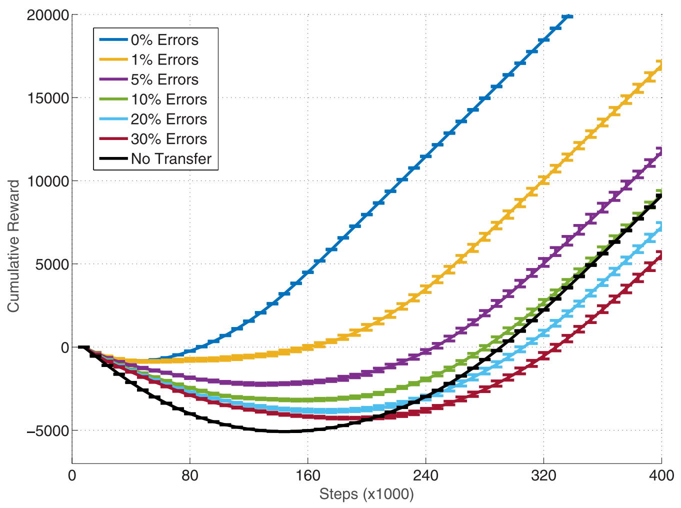

Fig. 6. Effect of prediction errors on cumulative reward in test environment #1. While the simulated errors significantly affected the time to convergence, the cost of learning was always lower than that in the no-knowledge-transfer case. Error rates here represent the number of errors, rather than their magnitude.

cost of learning, and the asymptotic slope represents the quality of the learned solution. The algorithm has converged on a solution at the point where the curve converges to its asymptotic slope. An RL algorithm has been categorically outperformed if its cumulative reward plot is everywhere dominated by that of another algorithm.

Our method of knowledge transfer using predictions afforded significant advantages in terms of both time-to-convergence (the number of steps spent in the environment before the final solution is discovered, as evidenced by a stabilization of the slope of the cumulative reward plot), and cost of learning (the largest negative cumulative reward reached during learning). In these tests, transfer by shared features offered no significant benefits over the baseline no-knowledge-transfer case. The reason for this will be considered in Section IV.

*3) Sensitivity to Prediction Accuracy:* In practical applications, it may be impossible to predict $P(s'|a,s)$ and $r$ with perfect accuracy. We thus tested the algorithm's sensitivity to imperfect predictions in the first test environment (Fig. 2) by replacing the agent-centric system with a predictor that generated perfect $<\hat{s}', \hat{r}>$ predictions with a specified probability, and random predictions otherwise. The random prediction included a random reward $\hat{r} \sim U(-1, 0)$ and a state $\hat{s'}$ randomly selected from one of the 9 states in a 3-by-3 box centered at the agent's current location. Fig. 6 shows the effects of 1%, 5%, 10%, 20%, and 30% random predictions on cumulative reward. Note that these error rates represent the *number* rather than the *magnitude* of the simulated prediction errors; since the errors themselves were randomly generated, some of them are catastrophic errors (e.g., that moving into a wall is possible, or that moving into the goal state is not), which causes the agent to behave in highly suboptimal ways. Thus, this test likely represents a worst case analysis.

Injecting prediction errors caused a significant drop in the cumulative reward over 400 000 steps. The effect on performance grew with the prediction error rate, with the difference between 0% and 1% error being the most striking. Error rates above 10% caused the agent to require more time than the no-knowledge-transfer case to converge on the optimum solution. However, even with error rates of 20% or 30%, the agent

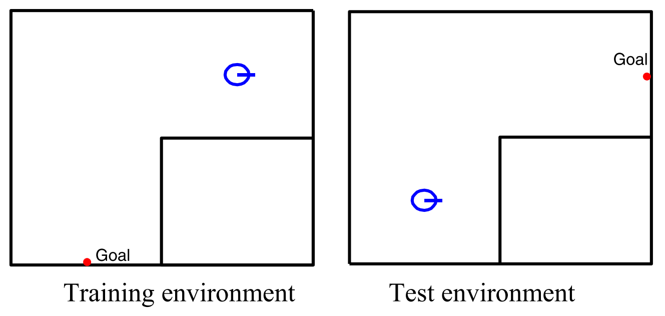

Fig. 7. Training and test environments used with the simulated Create robot. The robot must learn to navigate to a colored beacon while avoiding walls.

converged to the optimal solution with a lower cost of learning than the no-knowledge-transfer case.

## B. Example 2: Simulated Robot Control

Although the grid worlds in the previous section provide a straightforward problem with illustrative value, their state transitions are purely deterministic and perfectly predictable, making them somewhat unrealistic. We next tested the feasibility of our approach in a more realistic problem wherein transitions are stochastic and not perfectly predictable given the agent's sensor data, and wherein prediction errors would be more realistic than those simulated in the previous sensitivity analysis.

We tested our algorithm on a simulated iRobot Create robot in a simple navigation learning task. This test used the create robot simulator (obtained from https://sourceforge.net/projects/createsim) to simulate a 4 m × 4 m L-shaped enclosure (Fig. 7). The enclosure was discretized into 15 $x$-coordinates, 15 $y$-coordinates, and 8 robot orientations $\theta$, and the robot was allowed four actions corresponding to moving forward, moving backward, turning left, and turning right. It received a reward of $-0.1$ if it collided with a wall and $-0.05$ for each action otherwise. If it touched a colored goal beacon located on one of the enclosure walls, it received a reward of $+10$ and was relocated to the start state for another trial. The robot was allowed to learn a good agent-centric prediction model by solving the training problem 10 times before being tested for 1 hour in the test environment. This learning task is an uncomplicated problem in a simple simulated environment; but since training and test tasks differ only in the location of goal beacon, it effectively demonstrates how our knowledge transfer method can facilitate learning multiple tasks in the same environment.

Here, the environment-centric state variables included the robot's $x$ and $y$ coordinates and orientation $\theta$. The agent-centric state included four distance-to-wall readings from four distance sensors oriented at 0°, 90°, 180°, and 270°, and a Boolean value indicating whether the goal was currently in view of a forward-facing camera.

It is noteworthy that because this test involves a discretization of a continuous state space, there is inherent stochasticity in the transitions, which makes them impossible to predict with perfect accuracy. Because the state space is discretized coarsely, two spatial locations for which action outcomes are quite different can sometimes be interpreted as a single discretized state: from the agent's perspective, it seems as though action outcomes from this state are stochastic.

When the agent confuses agent-centric states (either due to the discretization effect, or because four distance sensors provide imperfect sensation of the environment) the agent-centric model may produce a $<s, a, \hat{s}', \hat{r}>$ prediction which—though correct for the supposed agent-centric state—is incorrect for the actual agent-centric state. These erroneous predictions are an example of *negative transfer*, wherein inappropriately transferred knowledge impedes learning.

The predictability of transitions is further degraded by the low quality sensor data (four distance readings are often not informative enough to predict collisions). This is in contrast to the grid world setting, in which transitions are deterministic and easily predicted based on the types of elements observed in the agent-centric space. This test thus demonstrates the applicability of our approach to situations where predictions cannot be made with perfect accuracy.

*1) Algorithm Implementation:* The implementation of Algorithm 1 for this setting used model-based RL as the environment-centric learning algorithm [57]. The model-based algorithm maintained a table of $Q$ values, as well as a table T[*s,a,s'*] to track the number of times action $a$ from state $s$ resulted in a transition to $s'$, allowing transition probabilities to be estimated as

$$P(s^{'}|s,a) = T[s,a,s']/$$
$$sum_{s_x} T[s,a,s_x].$$

Another table R[*s,a,s'*] tracked the average reward associated with the same transition. $Q$ values may then be calculated as

$$Q[s,a] = \sum_{s'} P(s,a,s')(R(s,a,s') + \gamma \max_{a'} Q[s^{'}, a']). \quad (4)$$

Given a state transition < *s,a,s',r* >, up to 200 $Q$ values were updated according to (4); the $Q$ values to be updated were selected using the prioritized sweeping algorithm [55]. These additional updates allow model-based RL algorithms to learn faster with less experience than model-free algorithms like $Q$ learning. This advantage is important for applications to robots and other real-world systems.

The agent-centric system used four $Q$-learning algorithms as described previously to predict $\Delta x$, $\Delta y$, $\Delta\theta$, and $r$. The knowledge transfer predicate was defined as in Section III-A1) with $t_a = 5$ and $t_e = 1$. The parameter $\gamma$ was set to 0.9.

The transfer by shared features approach was implemented using the same model-based environment-centric algorithm as our method, and the same agent-centric learning system as in Example 1. The transfer by abstract policy approach used the same model-based environment-centric algorithm as our method.

*2) Results:* The test results are shown in Fig. 8. The predictions generated by the agent-centric system during the test were only 74% accurate, but still allowed the agent to converge to a solution in less time than the no-knowledge-transfer case. Transfer by abstract policy allowed the agent

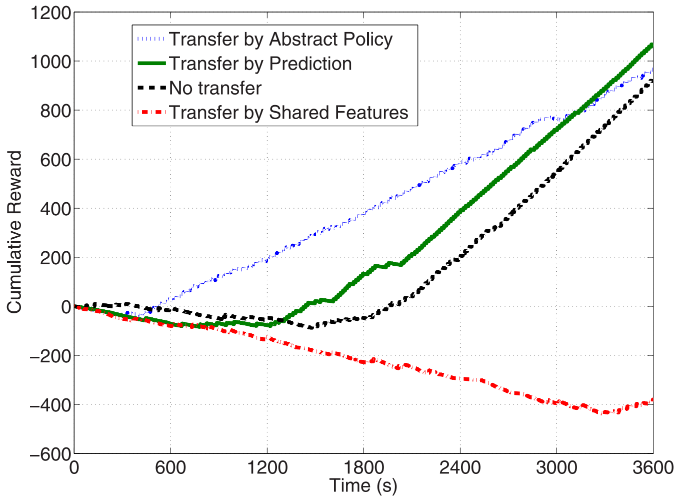

Fig. 8. Cumulative reward obtained by the robot in the test environment, using the various knowledge transfer methods.

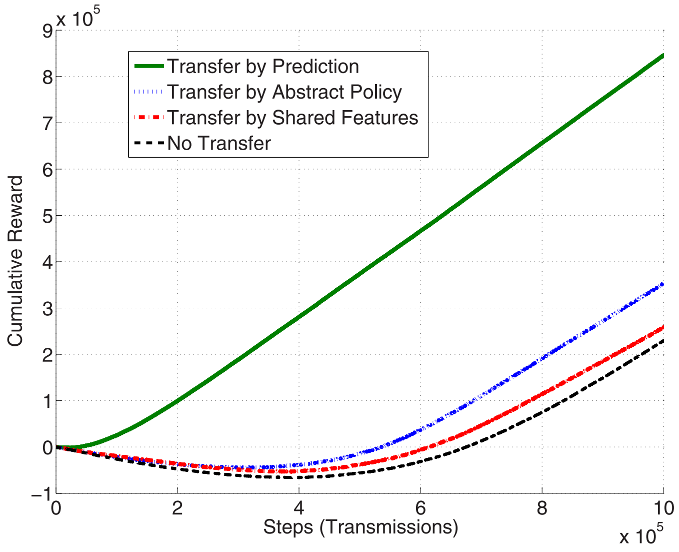

Fig. 9. Cumulative reward obtained in the network routing problem, using the various knowledge transfer methods.

to converge to a solution even more quickly, but the solution was highly suboptimal (in this case, the abstract policy did not represent the optimal policy for the new environment). Transfer by shared features was more hindrance than help; these results, as well as those from the grid world tests, show that knowledge transfer by shared features can fail in these navigation-style problems. An explanation for this will be put forward in Section IV.

### *C. Example 3: Transmission Routing in a Wireless Mesh Network*

We have thus far focused on navigation-style problems where agent-centric and environment-centric spaces are perhaps easiest understood, and become synonymous with egocentric and allocentric space. Our final example applies the proposed approach to a problem with no spatial navigation component.

We consider a network routing problem in which a number of wireless sensor nodes are distributed across a large area. Each node has a wireless link to several neighboring nodes, but must "hop" messages through the network to communicate with nodes beyond them. The quality of wireless links between node pairs varies, and the cost (i.e., the time and/or transmission power required) for a node to transmit to one of its neighbors is a random variable parameterized by the link quality. The task is to learn the lowest cost routing paths between each pair of nodes.

To simulate this problem, we used the Havel-Hakimi algorithm [58] to create a network of 64 nodes, each connected to five neighbors. Each connection was assigned a randomly generated quality indicator $\beta$, and the simulated cost of sending a transmission through that connection was a random value drawn from the beta distribution Beta $(1, \beta)$. It is assumed that the connections and link quality values are known *a priori*, but that optimal routes must be learned as messages are passed within the network. At each transmission, the reward received is the negative of the transmission cost. There is an additional $+10$ reward when a message arrives at its intended destination.

*1) Algorithm Implementation:* Here the environment-centric state consists of the node at which a message currently resides, and the intended destination node. The action set consists of transmitting the message from the current node to one of its five neighbors. The agent-centric state consists of the link quality values associated with those neighbors, and a Boolean indication of whether those neighbors are the destination node.

This test used a $Q$ learning implementation of the environment-centric learner, with $\alpha = 0.2$ and $\gamma = 0.9$. The agent-centric learner was a standard feedforward neural network implemented using the MATLAB neural network toolbox with one hidden layer of 10 neurons. This network was trained every 1000 steps on a history of the 10,000 most recent transitions. This network learned to predict reward for each transmission based on $\beta$ parameters (the task of predicting the next environment-centric state is trivial, since connections are known *a priori*). The knowledge-transfer predicate allowed predictions to be used after 1000 transmissions.

The transfer by shared features approach used the same $Q$-learning environment-centric and agent-centric learning algorithms as described in Example 1; the agent-centric algorithm discretized $\beta$ values into 50 bins to create a discrete state space on which to operate. The transfer by abstract policy approach also used the same $Q$-learning implementation described in Example 1, but when building an abstract policy it considered each action individually: identifying actions from other environment-centric states that shared the same (binned) $\beta$ parameter and were the optimal action from their environment-centric state.

*2) Results:* The cumulative reward obtained by each method is shown in Fig. 9. Transfer by shared features improves upon the no-knowledge-transfer case, while the more recent transfer by abstract policy provides further improvement. Transfer by prediction converged to the optimal solution the fastest.

## IV. Discussion

We have proposed in this paper an RL algorithm that includes both agent-centric and environment-centric learning

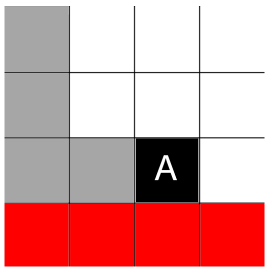

Fig. 10. Example grid world situation: the agent (marked as "A") is next to a lava hazard and has already explored the states shaded gray. Using predictions of actions' consequences, the agent can correctly assign high values to moving up and right, and lower values to moving left and down.

systems, and transfers knowledge between tasks in the form of predictions about actions' environmental consequences. The agent-centric learning system can generate these predictions in novel environments because agent-centric features maintain a consistent definition and constant semantics across tasks. These predictions can inform the environment-centric learning system of actions' effects before they are actually experienced, facilitating faster and lower cost learning in both model-free and model-based learning situations.

There is a subtle but important difference between our prediction-based approach and the existing approach of using agent-centric learning to provide shaping functions or initial $Q$ values: Shaping or *$Q$ value* initialization can provide the agent with a bias for or against executing certain actions from a given state, but tells the agent nothing about the consequences of those actions. Similarly, knowledge transfer via abstract policies identifies actions that were optimal in similar previous states, but not what the consequences of the actions will be. Consider, for example, an agent exploring a new grid world. Upon reaching the new environment-centric state illustrated in Fig. 10, an agent-centric learner might initialize $Q$ values for the "up," "right," and "left" actions to some low value (having learned that moving through open space carries a minor negative reward) and the $Q$ value for the "down" action to some large negative reward (having learned that moving into lava is undesirable). Similarly, an abstract policy might identify each of the up, right, and left actions as having been optimal in similar agent-centric states in the past. The agent is left with no information about what it *should* do, except perhaps that it should not move down. This could actually be worse than the no-knowledge-transfer case, in which an optimistic initialization would at least push the agent toward unexplored territory. In contrast, our proposed method allows the agent in Fig. 10 to predict the results as follows.

1) Executing the "left" action will move it left by one space (undesirable, because that space has already been explored).
2) Executing the "down" action will take it onto the lava with a large negative reward.
3) Executing the "up" and "right" actions will take it into novel states (desirable because the optimistic initialization assigns high value to the untried actions in those states).

Thus the agent-centric system's predictions inform the agent about the value of various actions, *given their consequences*.

We expect that our proposed method of using predictions to transfer knowledge will be advantageous whenever there are many actions to be executed before obtaining a reward, or when the agent-centric space does not directly indicate how the reward should be obtained. If, however, there are *few* actions to be executed before reaching a goal, or if the agent-centric space *does* specify the "direction" of the reward, then using $Q$ value initialization or abstract policies would be effective means of knowledge transfer as illustrated by [35] and [40].

Our proposal of using predictions for knowledge transfer builds upon the previous work in RL. In particular, it is an extension of the established fact that if a transition function is predictable (and therefore can be represented by a compact model) the computational complexity of RL can be reduced [50]. We also note that exploitation of predictability in an MDP is already widespread in the use of function approximation [8], which inherently assumes that $Q$ values can be predicted using a compact function of the state variables.

The efficacy of this proposed knowledge transfer method depends on the accuracy of the agent-centric system's predictions. We noted in Example 2 that our simulated robot had only four distance sensors, and that their instantaneous readings did not allow very accurate predictions. However, more accurate predictions may be possible if these sensors' readings *over time* could be used to create a more detailed map of obstacles (i.e., by a simultaneous localization and mapping system [59]). Thus, using predictions for knowledge transfer imposes no requirements on sensor cost or quality; only on prediction accuracy.

A few catastrophic prediction errors are likely more detrimental than constant, mild errors. To help prevent very bad predictions, the future work should connect our concept of the knowledge transfer predicate $\Psi$, to work on "knows what it knows" RL [60]–[62], a field of research that gives agents the ability to estimate when their experience with a particular state or action is sufficient to allow an action value to be calculated. The estimation is performed in a statistically sound way, so that the risk of calculating a value prematurely is arbitrarily minimized. Such methods could be used to determine when the agent-centric system can and cannot confidently provide a meaningful prediction, and could be an effective alternative to our simple $\Psi$ implementation.

Erroneous predictions can result from insufficient experience, inadequate agent-centric model capacity, or negative transfer. The first two problems can be addressed through the design of $\Psi$ and the agent-centric model, but a negative transfer may be a bigger challenge: Taylor and Stone [10] call it "one of the most troubling open questions" for transfer learning in RL. Negative transfer can occur in our framework whenever the agent confuses two agent-centric states, thus applying a prediction (which is correct in the source state) to a target state that has been misidentified. There is no trivial solution to this problem, but we note that this kind of negative transfer is precluded in the first experiment (Section III-A), because in that experiment the agent is allowed to sense the agent-centric state perfectly. Thus, avoiding negative transfer in our framework is at least partly an issue of sensor quality, and of agent-centric feature space design.

Our method facilitates knowledge transfer between tasks that share an agent-centric space, and our experiments demonstrate its advantages over other methods proposed for this particular situation. Transfer in other situations (between tasks that share a transition or reward function, for example) would be best accomplished using other methods, some of which are reviewed in [10]. Therefore, the use of predictions for knowledge transfer should ultimately be combined with other existing knowledge transfer methods. For example, our method could be used within the skill learning and transfer framework [14], or the state-abstraction scheme of [15] or [17]. Our framework continuously transfers small amounts of knowledge from the agent-centric to the environment-centric systems, but it could potentially be combined with a framework such as that used by Chaturvedi et al. [32], who used KL divergence [33] to identify appropriate times to transfer larger portions of a complete model. Indeed, biological agents probably achieve fast and flexible learning by employing a variety of knowledge transfer mechanisms in concert.

Another important direction for future research will be the generalization of our proposed method beyond the MDP model, to the partially observable MDP (POMDP) model. In the POMDP paradigm, the agent is not directly aware of its state, but instead makes observations from which it derives a belief over states. This paradigm could be advantageous in situations like that in Example 2, where our robot is not truly aware of its exact state. The principle of transferring knowledge by predicting state transitions could be generalized by instead predicting changes in the belief state. However, this may be best approached within an existing approximate POMDP solution framework [63]–[65]. We leave the algorithmic realization of this proposal to the future work.

Finally, the concept of learning from predictions opens up interesting possibilities for learning from extended predictions or predicted trajectories. For example, if a robot could predict multiple steps into the future, it could "explore" a large space instantly through predictions and update action values accordingly: this prediction capability could save enormous amounts of time if, for example, the space was represented as a finely discretized occupancy grid. Konidaris *et al.* [18], [35] have pointed out the value of learning macroactions or "options" in an agent-centric space. Learning from predictions presents a possibility of learning options as extended predictions with promising outcomes.

## V. Conclusion

As humans, we routinely and intuitively make predictions about the consequences of our actions. We can predict what will happen if we walk through a puddle instead of walking around, or the consequences of sleeping late on an important day at work. When we see an elevator, we predict that it will transport us to another floor in the same building. Predictions help guide our behavior and our learning in new situations.

We have proposed a new, prediction-based approach to knowledge transfer in RL, in which an agent-centric system learns to predict actions' environmental consequences. Since this agent-centric system can be applied to any scenario where its predictions are valid, our approach can provide an effective means of knowledge transfer and can accelerate RL in new scenarios. Using several types of test problems involving both agent-centric and environment-centric information, we have demonstrated that our approach can perform more effectively in such problems than existing alternatives.

## References

[1] J. Straddon, *Adaptive Behavior and Learning*. Cambridge, U.K.: Cambridge Univ. Press, 1983.

[2] R. S. Woodworth and E. L. Thorndike, "The influence of improvement in one mental function upon the efficiency of other functions. I," *Psychol. Rev.*, vol. 8, no. 3, pp. 247–261, 1901.

[3] E. L. Thorndike and R. S. Woodworth, "The influence of improvement in one mental function upon the efficiency of other functions. II. The estimation of magnitudes," *Psychol. Rev.*, vol. 8, no. 4, pp. 384–395, 1901.

[4] E. L. Thorndike and R. S. Woodworth, "The influence of improvement in one mental function upon the efficiency of other functions: III. Functions involving attention, observation and discrimination," *Psychol. Rev.*, vol. 8, no. 6, pp. 553–564, 1901.

[5] G. Salomon and D. N. Perkins, "Rocky roads to transfer: Rethinking mechanism of a neglected phenomenon," *Edu. Psychol.*, vol. 24, no. 2, pp. 113–142, 1989.

[6] A. L. Brown and M. J. Kane, "Preschool children can learn to transfer: Learning to learn and learning from example," *Cognit. Psychol.*, vol. 20, no. 4, pp. 493–523, Oct. 1988.

[7] S. Thrun and L. Pratt, *Learning to Learn*. New York, NY, USA: Springer, 2012.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[9] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[10] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Jul. 2009.

[11] T. G. Dietterich, "The MAXQ method for hierarchical reinforcement learning," in *Proc. 15th Int. Conf. Mach. Learn.*, San Mateo, CA, USA, 1998, pp. 118–126.

[12] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *Int. J. Robot. Res.*, vol. 32, no. 3, pp. 263–279, Mar. 2013.

[13] O. B. Kroemer, R. Detry, J. Piater, and J. Peters, "Combining active learning and reactive control for robot grasping," *Robot. Auton. Syst.*, vol. 58, no. 9, pp. 1105–1116, Sep. 2010.

[14] S. Hart and R. Grupen, "Learning generalizable control programs," *IEEE Trans. Auto. Mental Develop.*, vol. 3, no. 3, pp. 216–231, Sep. 2011.

[15] E. Chalmers, E. Bermudez Contreras, B. Robertson, A. Luczak, and A. Gruber, "Context-switching and adaptation: Brain-inspired mechanisms for handling environmental changes," in *Proc. Word Congr. Comput. Intell.*, Vancouver, BC, Canada, 2016, pp. 3522–3529.

[16] E. Chalmers, A. Luczak, and A. J. Gruber, "Computational properties of the hippocampus increase the efficiency of goal-directed foraging through hierarchical reinforcement learning," *Front. Comput. Neurosci.*, vol. 10, Dec. 2016, Art. no. 128.

[17] F. Shoeleh and M. Asadpour, "Graph based skill acquisition and transfer learning for continuous reinforcement learning domains," *Pattern Recognit. Lett.*, vol. 87, pp. 104–116, Feb. 2017.

[18] G. Konidaris and A. G. Barto, "Building portable options: Skill transfer in reinforcement learning.," in *Proc. IJCAI*, vol. 7. 2007, pp. 895–900.

[19] P. Y. Glorennec, "Fuzzy Q-learning and dynamical fuzzy Q-learning," in *Proc. 3rd IEEE Conf. Fuzzy Syst. World Congr. Comput. Intell.*, vol. 1. Jun. 1994, pp. 474–479.

[20] D.-H. Lee, I.-W. Park, and J.-H. Kim, "Q-learning using fuzzified states and weighted actions and its application to omni-direnctional mobile robot control," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom. (CIRA)*, Jun. 2009, pp. 102–107.

[21] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, "Continuous-state reinforcement learning with fuzzy approximation," in *Adaptive Agents and Multi-Agent Systems III, Adaptation and Multi-Agent Learning*. New York, NY, USA: Springer, 2008, pp. 27–43.

[22] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, "Approximate dynamic programming with a fuzzy parameterization," *Automatica*, vol. 46, no. 5, pp. 804–814, May 2010.
[23] K. Figueiredo, M. Vellasco, and M. A. Pacheco, "Hierarchical neuro-fuzzy models based on reinforcement learning for intelligent agents," in *Proc. Int. Work-Conf. Artif. Neural Netw.*, 2005, pp. 424–432.
[24] M. F. Corrêa, M. Vellasco, and K. Figueiredo, "Multi-agent systems with reinforcement hierarchical neuro-fuzzy models," *Auton. Agents Multi-Agent Syst.*, vol. 28, no. 6, pp. 867–895, Nov. 2014.
[25] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auto. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.
[26] K. Conn and R. A. Peters, "Reinforcement learning with a supervisor for a mobile robot in a real-world environment," in *Proc. Int. Symp. Comput. Intell. Robot. Autom.*, 2007, pp. 73–78.
[27] G. Wang, Z. Fang, P. Li, and B. Li, "Transferring knowledge from human-demonstration trajectories to reinforcement learning," *Trans. Inst. Meas. Control*, p. 142331216649655, Sep. 2016, doi: 10.1177/0142331216649655.
[28] M. S. Erden and K. Leblebicioğlu, "Free gait generation with reinforcement learning for a six-legged robot," *Robot. Auton. Syst.*, vol. 56, no. 3, pp. 199–212, Mar. 2008.
[29] N. Birdwell, S. Livingston, and I. Elhanany, "Reinforcement learning in sensor-guided aibo robots," Univ. Tennesse, Knoxville, TN, USA, Tech. Rep., 2007.
[30] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 2011, pp. 3828–3834.
[31] E. M. de Cote, E. O. Garcia, and E. F. Morales, "Transfer learning by prototype generation in continuous spaces," *Adapt. Behavior*, vol. 24, no. 6, pp. 464–478, 2016.
[32] I. Chaturvedi, Y.-S. Ong, and R. V. Arumugam, "Deep transfer learning for classification of time-delayed Gaussian networks," *Signal Process.*, vol. 110, pp. 250–262, May 2015.
[33] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.
[34] E. Wiewiora. (Jun. 2011). "Potential-based shaping and Q-value initialization are equivalent." [Online]. Available: https://arxiv.org/abs/1106.5267
[35] G. Konidaris, I. Scheidwasser, and A. G. Barto, "Transfer in reinforcement learning via shared features," *J. Mach. Learn. Res.*, vol. 13, pp. 1333–1371, May 2012.
[36] B. Banerjee and P. Stone, "General game learning using knowledge transfer," in *Proc. IJCAI*, 2007, pp. 672–677.
[37] W. Guofang, F. Zhou, L. Ping, and L. Bo, "Shaping in reinforcement learning via knowledge transferred from human-demonstrations," in *Proc. 34th Chin. Control Conf. (CCC)*, 2015, pp. 3033–3038.
[38] A. Mousavi, B. N. Araabi, and M. N. Ahmadabadi, "Context transfer in reinforcement learning using action-value functions," *Comput. Intell. Neurosci.*, vol. 2014, p. 52, Jan. 2014.
[39] A. Mousavi, B. N. Araabi, and M. N. Ahmadabaadi, "Context transfer and Q-transferable tasks," in *Proc. Workshops 29th AAAI Conf. Artif. Intell.*, 2015, pp. 30–34.
[40] V. Freire and A. H. R. Costa, "Comparative analysis of abstract policies to transfer learning in robotics navigation," in *Proc. Workshops 29th AAAI Conf. Artif. Intell.*, 2015, pp. 9–15.
[41] M. L. Koga, V. Freire, and A. H. R. Costa, "Stochastic abstract policies: Generalizing knowledge to improve reinforcement learning," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 77–88, Jan. 2015.
[42] G. Barth-Maron, D. Abel, J. MacGlashan, and S. Tellex, "Affordances as transferable knowledge for planning agents," in *Proc. AAAI Fall Symp. Ser.*, 2014, pp. 2–8.
[43] A. M. Bastos, W. M. Usrey, R. A. Adams, G. R. Mangun, P. Fries, and K. J. Friston, "Canonical microcircuits for predictive coding," *Neuron*, vol. 76, no. 4, pp. 695–711, Nov. 2012.
[44] L. H. Arnal and A.-L. Giraud, "Cortical oscillations and sensory predictions," *Trends Cognit. Sci.*, vol. 16, no. 7, pp. 390–398, 2012.
[45] R. A. Rescorla, "Simultaneous and successive associations in sensory preconditioning." *J. Experim. Psychol., Animal Behavior Process.*, vol. 6, no. 3, p. 207, 1980.
[46] M. Jeannerod, "Action monitoring and forward control of movements," in *The Handbook of Brain Theory and Neural Networks*, 2nd ed. Cambridge, MA, USA: MIT Press, 2003, pp. 83–84.
[47] P. M. Pilarski, T. B. Dick, and R. S. Sutton, "Real-time prediction learning for the simultaneous actuation of multiple prosthetic joints," in *Proc. IEEE Int. Conf. Rehabil. Robot. (ICORR)*, Jul. 2013, pp. 1–8.
[48] M. Gregor and J. Spalek, "Novelty detector for reinforcement learning based on forecasting," in *Proc. IEEE 12th Int. Symp. Appl. Mach. Intell. Inf. (SAMI)*, Oct. 2014, pp. 73–78.
[49] J. Modayil, A. White, and R. S. Sutton, "Multi-timescale nexting in a reinforcement learning robot," *Adapt. Behavior*, vol. 22, no. 2, pp. 146–160, 2014.
[50] M. Kearns and D. Koller, "Efficient reinforcement learning in factored MDPs," in *Proc. IJCAI*, vol. 16. 1999, pp. 740–747.
[51] C. Lemke, M. Budka, and B. Gabrys, "Metalearning: A survey of trends and technologies," *Artif. Intell. Rev.*, vol. 44, no. 1, pp. 117–130, Jun. 2015.
[52] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Mach. Learn.*, vol. 49, nos. 2–3, pp. 209–232, 2002.
[53] S. Koenig and R. G. Simmons, *Complexity Analysis of Real-Time Reinforcement Learning Applied to Finding Shortest Paths in Deterministic Domains*, DTIC document CMU-CS-93-106, 1992.
[54] R. I. Brafman and M. Tennenholtz, "R-max-a general polynomial time algorithm for near-optimal reinforcement learning," *J. Mach. Learn. Res.*, vol. 3, pp. 213–231, Oct. 2002.
[55] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Mach. Learn.*, vol. 13, no. 1, pp. 103–130, 1993.
[56] R. S. Sutton *et al.*, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *Proc. 10th Int. Conf. Auto. Agents Multiagent Syst.*, vol. 2. 2011, pp. 761–768.
[57] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proc. 7th Int. Conf. Mach. Learn.*, 1990, pp. 216–224.
[58] S. L. Hakimi, "On realizability of a set of integers as degrees of the vertices of a linear graph. I," *J. Soc. Ind. Appl. Math.*, vol. 10, no. 3, pp. 496–506, 1962.
[59] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.
[60] L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl, "Knows what it knows: A framework for self-aware learning," *Mach. Learn.*, vol. 82, no. 3, pp. 399–443, Nov. 2010.
[61] A. L. Strehl and M. L. Littman, "Online linear regression and its application to model-based reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1417–1424.
[62] E. Brunskill, B. R. Leffler, L. Li, M. L. Littman, and N. Roy, "Provably efficient learning with typed parametric models," *J. Mach. Learn. Res.*, vol. 10, pp. 1955–1988, Aug. 2009.
[63] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Auto. Agents Multi-Agent Syst.*, vol. 27, no. 1, pp. 1–51, Jul. 2013.
[64] A. Foka and P. Trahanias, "Real-time hierarchical POMDPs for autonomous robot navigation," *Robot. Auto. Syst.*, vol. 55, no. 7, pp. 561–571, Jul. 2007.
[65] D. K. Grady, M. Moll, and L. E. Kavraki, "Extending the applicability of POMDP solutions to robotic tasks," *IEEE Trans. Robot.*, vol. 31, no. 4, pp. 948–961, Aug. 2015.

**Eric Chalmers** received the B.Sc. degree in electrical engineering and the Ph.D. degree in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada.

He was a Post-Doctoral Fellow with the Department of Neuroscience, University of Lethbridge, Lethbridge, AB, Canada, where he developed brain-inspired machine learning techniques. He is currently a Lead Data Scientist with Farmers Edge, Lethbridge, AB, Canada, where he develops machine-learning-based decision support systems for precision agriculture.

**Edgar Bermudez Contreras** received the Ph.D. degree in computer science and artificial intelligence from the University of Sussex, Sussex, U.K., in 2010.

He is currently a Post-Doctoral Research Fellow with the University of Lethbridge, Lethbridge, AB, Canada. His current research interests include from computational neuroscience to machine learning.

**Brandon Robertson** is currently pursuing the B.Sc. degree in computer science with the University of Lethbridge, Lethbridge, AB, Canada.

His current interests include research revolves around using reinforcement learning.

**Artur Luczak** received the M.Sc. degree in biomedical engineering from the Wroclaw University of Technology, Wroclaw, Poland, and the Ph.D. degree from the Jagiellonian University Medical College, Krakow, Poland, complemented by fellowships in The Netherlands (Huygens Scholarship), Italy (Marie Curie Fellowship), and France.

He was a Post-Doctoral Fellow at Yale University, New Haven, CT, USA. He studied neural information processing using experimental and theoretical methods with Rutgers University, Middlesex County, NJ, USA. He is currently an Assistant Professor with the Canadian Center for Behavioral Neuroscience, University of Lethbridge, Lethbridge, AB, Canada, and a Visiting Associate Professor with Stanford University, Stanford, CA, USA. His laboratory investigates spontaneous interactions between neurons, and how these relations are distorted in different neurological disorders such as epilepsy.

Dr. Luczak was elected to the College of the Royal Society of Canada for his work in 2016.

**Aaron Gruber** received the Ph.D. degree in biomedical engineering from Northwestern University, Evanston, IL, USA.

His current research interests include discovering how the brain processes information to generate decisions, and how conferring brain-such as function to artificial neural networks can produce superior performance.