

# Transfer Learning in Deep Reinforcement Learning: A Survey

Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain, and Jiayu Zhou

**Abstract**—Reinforcement learning is a learning paradigm for solving sequential decision-making problems. Recent years have witnessed remarkable progress in reinforcement learning upon the fast development of deep neural networks. Along with the promising prospects of reinforcement learning in numerous domains such as robotics and game-playing, transfer learning has arisen to tackle various challenges faced by reinforcement learning, by transferring knowledge from external expertise to facilitate the efficiency and effectiveness of the learning process. In this survey, we systematically investigate the recent progress of transfer learning approaches in the context of deep reinforcement learning. Specifically, we provide a framework for categorizing the state-of-the-art transfer learning approaches, under which we analyze their goals, methodologies, compatible reinforcement learning backbones, and practical applications. We also draw connections between transfer learning and other relevant topics from the reinforcement learning perspective and explore their potential challenges that await future research progress.

**Index Terms**—Transfer Learning, Reinforcement Learning, Deep Learning, Survey.

## 1 INTRODUCTION

Reinforcement Learning (RL) is an effective framework to solve sequential decision-making tasks, where a learning agent interacts with the environment to improve its performance through trial and error [1]. Originated from cybernetics and thriving in computer science, RL has been widely applied to tackle challenging tasks which were previously intractable. Traditional RL algorithms were mostly designed for tabular cases, which provide principled solutions to simple tasks but face difficulties when handling highly complex domains, *e.g.* tasks with 3D environments. With the recent advances in computing capability and deep learning research, the combination of RL agents and deep neural networks is developed to address challenging tasks. The combination of deep learning with RL is hence referred to as *Deep Reinforcement Learning* [2], which learns powerful function approximators using deep neural networks to address complicated domains. RL powered with deep learning has achieved notable success in applications such as robotics control [3, 4] and game playing [5]. It also has a promising prospects in domains such as health informatics [6], electricity networks [7], intelligent transportation systems[8, 9], to name just a few.

Besides its remarkable advancement, RL still faces intriguing difficulties induced by the exploration-exploitation dilemma [1]. Specifically, for practical RL problems, the environment dynamics are usually unknown, and the agent cannot exploit knowledge about the environment to improve its performance until enough interaction experiences are collected via exploration. Due to the partial observability, sparse feedbacks, and the high complexity of state and action spaces, acquiring sufficient interaction samples can be prohibitive, which may even incur safety concerns for

domains such as automatic-driving and health informatics, where the consequences of wrong decisions can be too high to take. The abovementioned challenges have motivated various efforts to improve the current RL procedure. As a result, *transfer learning*, or equivalently referred as *knowledge transfer*, which is a technique to utilize external expertise from other domains to benefit the learning process of the target task, becomes a crucial topic in RL.

While transfer learning techniques have been extensively studied in the *supervised learning* domain [10], it is still an emerging topic for RL. Transfer learning can be more complicated under the RL framework, in that the knowledge needs to transfer in the context of a Markov Decision Process. Moreover, due to the delicate components of the Markov decision process, expert knowledge may take different forms, which need to transfer in different ways. Noticing that previous efforts on summarizing transfer learning for the RL domain have not covered the most recent advancement [11, 12], in this survey, we make a comprehensive investigation of *Transfer Learning in Deep Reinforcement Learning*. Especially, we propose a systematic framework to categorize the state-of-the-art transfer learning techniques into different sub-topics, review their theories and applications, and analyze their inter-connections.

The rest of this survey is organized as follows: In section 2, we introduce the preliminaries of RL and its key algorithms, including those recently designed based on deep neural networks. Next, we clarify the definition of transfer learning in the context of RL and discuss its relevant research topics (Section 2.4). In Section 3, we provide a framework to categorize transfer learning approaches from multiple perspectives, analyze their fundamental differences, and summarize their evaluation metrics (Section 3.3). In Section 4, we elaborate on different transfer learning approaches in the context of deep RL, organized by the format of transferred knowledge, such as *reward shaping* (Section 4.1), *learning from demonstrations* (Section 4.2), or *learning from*

• Zhuangdi Zhu, Anil K. Jain, and Jiayu Zhou are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, 48824. E-mail: {zhuangdi.zhu, jain, jiayuz}@msu.edu

• Kaixiang Lin is with the Amazon Alexa AI. E-mail: lkxcarson@gmail.com

teacher policies (Section 4.3). We also investigate transfer learning approaches by the way that knowledge transfer occurs, such as inter-task mapping (Section 4.4), or learning transferrable representations (Section 4.5), etc. We discuss the recent applications of transfer learning in the context of deep RL in Section 5 and provide some future perspectives and open questions in Section 6.

## 2 DEEP REINFORCEMENT LEARNING AND TRANSFER LEARNING

In this section, we provide a brief overview of the recent development in RL and the definitions of some key terminologies. Next, we provide categorizations to organize different transfer learning approaches, then point out some of the other topics in the context of RL, which are relevant to transfer learning but will not be elaborated on in this survey.

### 2.1 Reinforcement Learning Definitions

A typical RL problem can be considered as training an agent to interact with an environment that follows a **Markov Decision Process (MPD)** [13]. For each interaction with the MDP, the agent starts with an initial state and performs an action accordingly, which yields a reward to guide the agent actions. Once the action is taken, the MDP transits to the next state by following the underlying transition dynamics of the MDP. The agent accumulates the time-discounted rewards along with its interactions with the MDP. A subsequence of interactions is referred to as an *episode*. For MDPs with infinite horizons, one can assume that there are *absorbing states*, such that any action taken upon an absorbing state will only lead to itself and yield zero rewards. All above-mentioned components in an MDP can be represented using a tuple, *i.e.*  $\mathcal{M} = (\mu_0, \mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R}, \mathcal{S}_0)$ , in which:

- $\mu_0$  is the set of *initial states*.
- $\mathcal{S}$  is the *state space*.
- $\mathcal{A}$  is the *action space*.
- $\mathcal{T}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  is the *transition probability distribution*, where  $\mathcal{T}(s'|s, a)$  specifies the probability of the state transitioning to  $s'$  upon taking action  $a$  from state  $s$ .
- $\mathcal{R}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  is the *reward distribution*, where  $\mathcal{R}(s, a, s')$  is the reward that an agent can get by taking action  $a$  from state  $s$  with the next state being  $s'$ .
- $\gamma$  is a discounted factor, with  $\gamma \in (0, 1]$ .
- $\mathcal{S}_0$  is the set of *absorbing states*.

A RL agent behaves in  $\mathcal{M}$  by following its policy  $\pi$ , which is a mapping from states to actions:  $\pi: \mathcal{S} \rightarrow \mathcal{A}$ . For a stochastic policy  $\pi$ ,  $\pi(a|s)$  denotes the probability of taking action  $a$  from state  $s$ . Given an MDP  $\mathcal{M}$  and a policy  $\pi$ , one can derive a *value function*  $V_{\mathcal{M}}^{\pi}(s)$ , which is defined over the state space:  $V_{\mathcal{M}}^{\pi}(s) = \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots; \pi, s]$ , where  $r_i = \mathcal{R}(s_i, a_i, s_{i+1})$  is the reward that an agent receives by taking action  $a_i$  in the  $i$ -th state  $s_i$ , and the next state transits to  $s_{i+1}$ . The expectation  $\mathbb{E}$  is taken over  $s_0 \sim \mu_0, a_i \sim \pi(\cdot|s_i), s_{i+1} \sim \mathcal{T}(\cdot|s_i, a_i)$ . The value function estimates the *quality* of being in state  $s$ , by evaluating the expected rewards that an agent can get from  $s$ , given that the agent follows policy  $\pi$  in the environment  $\mathcal{M}$  afterward. Similar to the value function, each policy also carries a *Q-function*, which is defined over the state-action

space to estimate the quality of taking action  $a$  from state  $s$ :  $Q_{\mathcal{M}}^{\pi}(s, a) = \mathbb{E}_{s' \sim \mathcal{T}(\cdot|s, a)} [\mathcal{R}(s, a, s') + \gamma V_{\mathcal{M}}^{\pi}(s')]$ . The objective for a RL agent is to learn an optimal policy  $\pi_{\mathcal{M}}^*$  to maximize the expectation of accumulated rewards, so that:  $\forall s \in \mathcal{S}, \pi_{\mathcal{M}}^*(s) = \arg \max_{a \in \mathcal{A}} Q_{\mathcal{M}}^*(s, a)$ , where  $Q_{\mathcal{M}}^*(s, a) = \sup_{\pi} Q_{\mathcal{M}}^{\pi}(s, a)$ .

### 2.2 Reinforcement Learning Algorithms

In this section, we review the key RL algorithms developed over the recent years, which provide cornerstones for the transfer learning approaches discussed in this survey.

**Prediction and Control:** an RL problem can be disassembled into two subtasks: *prediction* and *control* [1]. In the *prediction* phase, the quality of the current policy is being evaluated. In the *control* phase or the *policy improvement* phase, the learning policy is adjusted based on evaluation results from the *prediction* step. Policies can be improved by iteratively conducting these two steps. The above procedure is therefore called *policy iteration*.

Policy iterations can be *model-free*, which means that the target policy is optimized without requiring knowledge of the MDP transition dynamics. Traditional model-free RL includes **Monte-Carlo** methods, which uses *samples of episodes* to estimate the value of each state based on complete episodes starting from that state. Monte-Carlo methods can be *on-policy* if the samples are collected by following the target policy, or *off-policy* if the episodic samples are collected by following a *behavior policy* that is different from the target policy.

**Temporal Difference (TD) Learning** is an alternative to Monte-Carlo for solving the *prediction* problem. The key idea behind TD-learning is to learn the state quality function by *bootstrapping*. It can also be extended to solve the *control* problem so that both value function and policy can get improved simultaneously. Examples of *on-policy* TD-learning algorithms include SARSA [14], Expected SARSA [15], Actor-Critic [16], and its deep neural network extension called A3C [17]. The *off-policy* TD-learning approaches include SAC [18] for continuous state-action spaces, and *Q*-learning [19] for discrete state-action spaces, along with its variants built on deep-neural networks, such as DQN [20], Double-DQN [20], Rainbow [21], etc.

TD-learning approaches, such as *Q*-learning, focus more on estimating the state-action value functions. **Policy Gradient**, on the other hand, is a mechanism that emphasizes on direct optimization of a parametrizable policy. Traditional policy-gradient approaches include REINFORCE [22]. Recent years have witnessed the joint presence of TD-learning and policy-gradient approaches. Representative algorithms along this line include Trust region policy optimization (TRPO) [23], Proximal Policy optimization (PPO) [24], Deterministic policy gradient (DPG) [25] and its extensions such as DDPG [26] and Twin Delayed DDPG [27].

### 2.3 Transfer Learning in the Context of Reinforcement Learning

**Remark 1.** Without losing clarity, for the rest of this survey, we refer to MDPs, domains, and tasks equivalently.

We now provide a concrete definition of transfer learning from the RL perspective:

**Remark 2. [Transfer Learning in the Context of RL]** Given a set of *source domains*  $\mathcal{M}_s = \{\mathcal{M}_s | \mathcal{M}_s \in \mathcal{M}_s\}$  and a *target domain*  $\mathcal{M}_t$ , Transfer Learning aims to learn an optimal policy  $\pi^*$  for the target domain, by leveraging exterior information  $\mathcal{I}_s$  from  $\mathcal{M}_s$  as well as interior information  $\mathcal{I}_t$  from  $\mathcal{M}_t$ , where:

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{s \sim \mu_0^t, a \sim \pi} [Q_{\mathcal{M}}^\pi(s, a)],$$

where  $\pi = \phi(\mathcal{I}_s \sim \mathcal{M}_s, \mathcal{I}_t \sim \mathcal{M}_t) : \mathcal{S}^t \rightarrow \mathcal{A}^t$  is a function mapping from the states to actions for the target domain  $\mathcal{M}_t$ , which is learned based on information from both  $\mathcal{I}_t$  and  $\mathcal{I}_s$ .

In the above definition, we use  $\phi(\mathcal{I})$  to denote the learned policy based on information  $\mathcal{I}$ . Especially, in the context of deep RL, the policy  $\pi$  is learned using deep neural networks. For the simplistic case, knowledge can transfer between two agents within the same domain, which results in  $|\mathcal{M}_s| = 1$ , and  $\mathcal{M}_s = \mathcal{M}_t$ . One can consider regular RL without transfer learning as a special case of the above definition, by treating  $\mathcal{I}_s = \emptyset$ , so that a policy  $\pi$  is learned purely on the feedback provided by the target domain, i.e.  $\pi = \phi(\mathcal{I}_t)$ .

## 2.4 Related Topics

In addition to transfer learning, other efforts have been made to benefit RL by leveraging different forms of supervision. In this section, we briefly discuss other techniques that are relevant to transfer learning by analyzing the differences and connections between transfer learning and these relevant techniques, which we hope can further clarify the scope of this survey.

*Imitation Learning* aims to train a policy to mimic the behavior of an expert policy. It is considered as an alternative to RL to solve sequential decision-making problems when the environment feedbacks are unavailable [28–30]. Closely related to transfer learning, *imitation learning* can be actually adapted into a transfer learning approach called *Learning from Demonstrations (LfD)*, which will be elaborated in Section 4.2. What distinguishes LfD and the classic *Imitation Learning* approaches is that LfD still interacts with the domain to access reward signals, in the hope of improving the target policy assisted by a few expert demonstrations, rather than recovering the reward functions or the expert behavior. LfD can be more effective than imitation learning when the expert demonstrations are actually sub-optimal [31, 32].

*Lifelong Learning*, or *Continual Learning*, refers to the ability to learn multiple tasks that are temporally or spatially related. The key to acquiring *Lifelong Learning* is a tradeoff between obtaining new information over time and retaining the previously learned knowledge across new tasks. *Lifelong Learning* is a technique that is applicable to both supervised learning [33] and RL [34, 35]. *Lifelong Learning* can be a more challenging task compared to transfer learning, mainly because that it requires an agent to transfer knowledge across a sequence of dynamically-changing tasks which cannot be foreseen, rather than performing knowledge transfer among a fixed group of tasks. Moreover, the ability of automatic task detection can also be a requirement for *Lifelong Learning* [36], whereas for transfer learning, the agent is usually notified of the emergence of a new task.

**Hierarchical RL** has been proposed to resolve real-world tasks that are hierarchical. Different from traditional RL, in a hierarchical RL setting, the action space is grouped into different granularities to form higher-level macro actions. Accordingly, the learning task is also decomposed into hierarchically dependent subgoals. Most well-known hierarchical RL frameworks include *Feudal learning* [37], *Options framework* [38], *Hierarchical Abstract Machines* [39], and *MAXQ* [40]. Given the higher-level abstraction on tasks, actions, and state spaces, hierarchical RL can facilitate knowledge transfer across similar domains.

**Multi-Agent RL** has strong connections with *Game Theory* [41]. Different from a single-agent setting, multi-agent RL considers an MDP with multiple agents acting simultaneously in the environment. It aims to solve problems that were difficult or infeasible to be addressed by a single RL agent [42]. The interactive mode for multiple agents can either be independent, cooperative, competitive, or even a hybrid setting [43]. Approaches of knowledge transfer for Multi-agent RL fall into two classes: inter-agent transfer and intra-agent transfer. We refer users to [44] for a more comprehensive survey under this problem setting. Different from their perspective, this survey emphasizes the general transfer learning approaches for a single agent scenario, although approaches mentioned in this survey may also be applicable to multi-agent MPDs.

## 3 ANALYZING TRANSFER LEARNING FROM MULTIPLE PERSPECTIVES

In this section, we provide multiple perspectives to analyze transfer learning approaches in the context of RL. We provide an illustrative example to discuss the potential differences in the source and target domain and the different forms of transferrable knowledge. Then we summarize the metrics for evaluating transfer learning approaches.

### 3.1 Categorization of Transfer Learning Approaches

We point out that transfer learning approaches can be categorized by answering the following key questions:

- 1) **What knowledge is transferred:** Knowledge from the source domain can take different forms of supervision, such as expert experiences [45], the action probability distribution of an expert policy [46], or even a potential function that estimates the quality of state and action pairs in the source or target MDP [47]. The divergence in the *representations* and *granularities* of knowledge fundamentally decides the way that transfer learning is performed. The *quality* of the transferred knowledge, e.g. whether it comes from an oracle policy [48] or is provided by a sub-optimal teacher [32], also affects the way transfer learning methods are designed.
- 2) **What RL frameworks are compatible with the transfer learning approach:** We can rephrase this question into other forms, e.g., *is the transfer learning approach policy-agnostic, or does it only apply to certain types of RL backbones, such as the Temporal Difference (TD) methods?* Answers to this question are closely related to the format of the transferred knowledge. For example, transferring knowledge in the form of expert demonstrations are usually policy-agnostic (see Section 4.2), while policy

distillation, as will be discussed in Section 4.3, may not be suitable for RL algorithms such as DQN, which does not explicitly learn a policy function.

- 3) **What is the difference between the source and the target domain:** Some transfer learning approaches are suitable for the scenario where the source domain  $\mathcal{M}_s$  and the target domain  $\mathcal{M}_t$  are equivalent, whereas others are designed to transfer knowledge between different domains. For example, in video gaming tasks where observations are RGB pixels,  $\mathcal{M}_s$  and  $\mathcal{M}_t$  may share the same action space ( $\mathcal{A}$ ) but differs in their observation spaces ( $\mathcal{S}$ ). For other problem settings, such as the goal-conditioned RL [49], the two domains may differ only by the reward distribution:  $\mathcal{R}_s \neq \mathcal{R}_t$ . Such domain difference induces difficulties in transfer learning and affects how much knowledge can transfer.
- 4) **What information is available in the target domain:** While the cost of accessing knowledge from source domains is usually affordable, it can be prohibitive for the learning agent to access the target domain, or the learning agent can only have a very limited number of environment interactions due to a high sampling cost. Examples for this scenario include learning an auto-driving agent after training it in simulated platforms [50], or training a navigation robot using simulated image inputs before adapting it to real environments [51]. The accessibility of information in the target domain can affect the way that transfer learning approaches are designed.
- 5) **How sample-efficient the transfer learning approach is:** This question is related to the previous one regarding the accessibility of a target domain. Compared with learning from scratch, transfer learning enables the learning agent with better initial performance, which usually needs few interactions with the target domain to converge to a good policy, guided by the transferred knowledge. Based on the number of interactions needed to enable transfer learning, we can categorize approaches into the following classes: (i) *Zero-shot* transfer, which learns an agent that is directly applicable to the target domain without requiring any training interactions with it; (ii) *Few-shot* transfer, which only requires a few samples (interactions) from the target domain; (iii) *Sample-efficient* transfer, where an agent can benefit by transfer learning to be more sample efficient compared to normal RL.
- 6) **What are the goals of transfer learning:** We can answer this question by analyzing two aspects of a transfer learning approach: (i) the evaluation metrics and (i) the objective function. Evaluation metrics can vary from the asymptotic performance to the training iterations used to reach a certain performance threshold, which implies the different emphasis of the transfer learning approach. On the other hand, transfer learning approaches may optimize towards various objective functions augmented with different regularizations, which is usually hinged on the format of the transferred knowledge. For example, maximizing the policy entropy can be combined with the maximum-return learning objective in order to encourage explorations when the transferred knowledge is imperfect demonstrations [52].

### 3.2 Case Analysis of Transfer Learning

In this section, we use *HalfCheetah*<sup>1</sup> as a working example to illustrate how transfer learning can be performed between the source and the target domain. *HalfCheetah* is a standard RL benchmark for solving physical locomotion tasks, whose objective is to train a two-leg agent to run as fast as possible without losing control of itself.

#### 3.2.1 Potential Domain Differences:

During transfer learning, the differences between the source and target domain may reside in any component of an MDP. For instance, domains for learning a *HalfCheetah* agent can be different in the following aspects:

- $\mathcal{S}$  (State-space): domains can be made different by extending or constraining the available positions for the *HalfCheetah* agent to move.
- $\mathcal{A}$  (Action-space) can be adjusted by changing the range of available torques for the thigh, shin, or foot of the agent.
- $\mathcal{R}$  (Reward function): a domain can be simplified by using only the distance moved forward as rewards or be perplexed by using the scale of accelerated velocity in each direction as extra penalty costs.
- $\mathcal{T}$  (Transition dynamics): two domains can differ by following different physical rules, leading to different transition probabilities given the same state-action pairs.
- $\mu_0$  (Initial states): the source and target domains may have different initial states, specifying where and with what posture the agent can start moving.
- $\tau$  (Trajectories): the source and target domains may allow a different number of steps for the agent to move before a task is done.

#### 3.2.2 Transferrable Knowledge:

We list the following transferrable knowledge, assuming that the source and target domains are variants of the *HalfCheetah* benchmark, although other forms of knowledge transfer may also be feasible:

- **Demonstrated trajectories:** the target agent can learn from the behavior of a pre-trained expert, e.g. a sequence of running demonstrations.
- **Model dynamics:** the learning agent may access an approximation model of the physical dynamics, which is learned from the source domain but also applicable in the target domain. The agent can therefore perform dynamic programming based on the physical rules, running as fast as possible while avoiding losing its control due to the accelerated velocity.
- **Teacher policies:** an expert policy may be consulted by the learning agent, which outputs the probability of taking different actions upon a given state example.
- **Teacher value functions:** besides teacher policy, the learning agent may also refer to the value function derived by a teacher policy, which implies the quality of state-actions from the teacher's point of view.

### 3.3 Evaluation metrics

In this section, we enumerate the following representative metrics for evaluating transfer learning approaches, some of which have also been summarized in prior work [11, 53]:

1. <https://gym.openai.com/envs/HalfCheetah-v2/>

- *Jumpstart performance* (*jp*): the initial performance (returns) of the agent.
- *Asymptotic performance* (*ap*): the ultimate performance (returns) of the agent.
- *Accumulated rewards* (*ar*): the area under the learning curve of the agent.
- *Transfer ratio* (*tr*): the ratio between *asymptotic performance* of the agent with transfer learning and *asymptotic performance* of the agent without transfer learning.
- *Time to threshold* (*tt*): the learning time (iterations) needed for the target agent to reach certain performance threshold.
- *Performance with fixed training epochs* (*pe*): the performance achieved by the target agent after a specific number of training iterations.
- *Performance sensitivity* (*ps*): the variance in returns using different hyper-parameter settings.

The above criteria mainly focus on the *learning process* of the target agent. In addition, we introduce the following metrics from the perspective of *transferred knowledge*, which, although commensurately important for evaluation, have not been explicitly discussed by prior art:

- *Necessary knowledge amount*: the necessary *amount* of the knowledge required for transfer learning in order to achieve certain performance thresholds. Examples along this line include the number of designed source tasks [54], the number of expert policies, or the number of demonstrated interactions [29] required to enable knowledge transfer.
- *Necessary knowledge quality*: the necessary *quality* of the knowledge required to enable effective transfer learning. This metric helps in answering questions such as (i) Does the transfer learning approach rely on near-oracle knowledge from the source domain, such as expert-level demonstrations/policies [28], or (ii) is the transfer learning technique feasible even given sub-optimal knowledge [32]?

Metrics from the perspective of transferred knowledge are more subtle because transfer learning approaches differ in various perspectives, including the forms of transferred knowledge, the RL frameworks utilized to enable such transfer, and the difference between the source and the target domain. It may lead to biased evaluations by comparing transfer learning approaches from just one viewpoint. However, we believe that explicating these knowledge-related metrics will help in designing more generalizable and efficient transfer learning approaches.

In general, most of the abovementioned metrics can be considered as evaluating two abilities of a transfer learning approach: the *mastery* and *generalization*. *Mastery* refers to how well the learned agent can ultimately perform in the target domain, while *generalization* refers to the ability of the learning agent to quickly adapt to the target domain assisted by the transferred knowledge. Metrics such as *asymptotic performance*, *accumulated rewards*, and *transfer ratio*, evaluates the ability of *mastery*, whereas metrics of *jumpstart performance*, *performance sensitivity*, *necessary knowledge amount* and *necessary knowledge quality* emphasizes more on the ability of *generalization*. Metrics such as *time to threshold*, for example, can measure either the *mastery* ability or the *generalization* ability, depending on the choice of different thresholds. A threshold approaching the optimal emphasizes

more on *mastery*, while a lower threshold may focus on the *generalization* ability. Equivalently, *performance with fixed training epochs* can also focus on either side depending on the choice of the number of training epochs.

## 4 TRANSFER LEARNING APPROACHES

In this section, we elaborate on various transfer learning approaches and organize them into different sub-topics, mostly by answering the question of “*what knowledge is transferred*”. For each type of transfer learning approach, we investigate them by following the other criteria mentioned in Section 3. We start with the *reward shaping* approach (Section 4.1), which is generally applicable to different RL algorithms while requiring minimal changes to the underline RL framework, and overlaps with the other transfer learning approaches discussed in this chapter. We also provide an overview of different transfer learning approaches discussed in this survey in Figure 1.

### 4.1 Reward Shaping

Reward Shaping (RS) is a technique that leverages the exterior knowledge to reconstruct the reward distribution of the target domain to guide the agent’s policy learning. More specifically, in addition to the environment reward signals, RS learns a reward-shaping function  $\mathcal{F} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  to render auxiliary rewards, provided that the additional rewards contain external knowledge to guide the agent for better action selections. Intuitively, an RS strategy will assign higher rewards to more beneficial state-actions, which can navigate the agent to desired trajectories. As a result, the agent will learn its policy using the newly shaped rewards  $\mathcal{R}' : \mathcal{R}' = \mathcal{R} + \mathcal{F}$ , which means that RS has altered the target domain with a different reward function:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R}) \rightarrow \mathcal{M}' = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R}').$$

Along the line of RS, *Potential based Reward Shaping* (PBRs) is one of the most classical approaches. [47] proposed PBRs to form a shaping function  $F$  as the difference between two *potential functions* ( $\Phi(\cdot)$ ):

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s), \quad (1)$$

where the potential function  $\Phi(\cdot)$  comes from the knowledge of expertise and evaluates the quality of a given state. It has been proved that, without further restrictions on the underlying MDP or the shaping function  $F$ , PBRs is sufficient and necessary to preserve the policy invariance. Moreover, the optimal  $Q$ -function in the original and transformed MDP are related by the potential function:

$$Q_{\mathcal{M}'}^*(s, a) = Q_{\mathcal{M}}^*(s, a) - \Phi(s), \quad (2)$$

which draws a connection between potential based reward-shaping and advantage-based learning approaches [55].

The idea of PBRs was extended to [56], which formulated the potential as a function over both the state and the action spaces. This approach is called *Potential Based state-action Advice* (PBA). The potential function  $\Phi(s, a)$  therefore evaluates how beneficial an action  $a$  is to take from state  $s$ :

$$F(s, a, s', a') = \gamma\Phi(s', a') - \Phi(s, a). \quad (3)$$

One limitation of PBA is that it requires on-policy learning, which can be sample-inefficient, as in Equation (3),  $a'$  is the

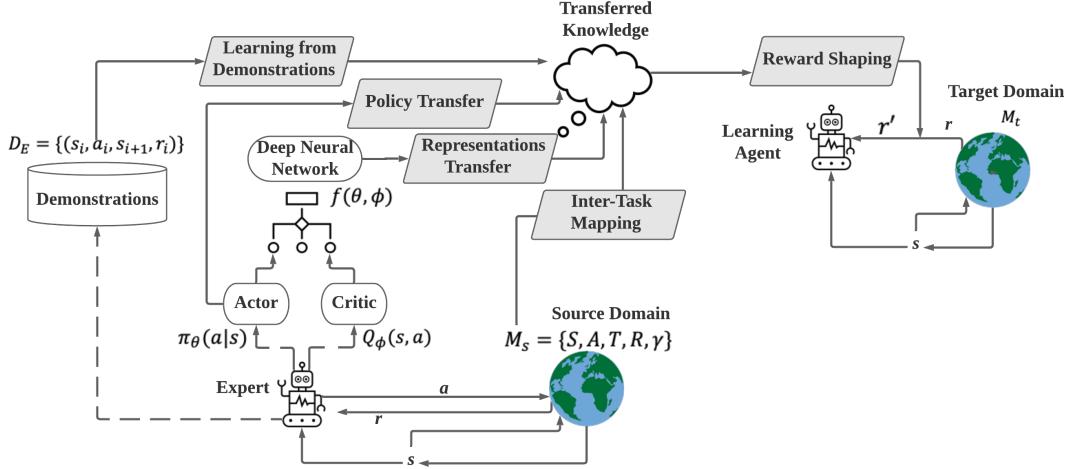


Fig. 1: An overview of different transfer learning approaches, organized by the format of transferred knowledge.

action to take upon state  $s$  is transitioning to  $s'$  by following the learning policy.

Traditional RS approaches assumed a static potential function, until [57] proposed a *Dynamic Potential Based (DPB)* approach which makes the potential a function of both states and time:  $F(s, t, s', t') = \gamma\Phi(s', t') - \Phi(s, t)$ . They proved that this dynamic approach can still maintain policy invariance:  $Q_{\mathcal{M}'}^*(s, a) = Q_{\mathcal{M}}^*(s, a) - \Phi(s, t)$ , where  $t$  is the current timestep. [58] later introduced a way to incorporate any prior knowledge into a dynamic potential function structure, which is called *Dynamic Value Function Advice (DPBA)*. The underline rationale of DPBA is that, given any extra reward function  $R^+$  from prior knowledge, in order to add this extra reward to the original reward function, the potential function should satisfy:

$$\gamma\Phi(s', a') - \Phi(s, a) = F(s, a) = R^+(s, a).$$

If  $\Phi$  is not static but learned as an extra state-action *Value* function overtime, then the Bellman equation for  $\Phi$  is :

$$\Phi^\pi(s, a) = r^\Phi(s, a) + \gamma\Phi(s', a').$$

The shaping rewards  $F(s, a)$  is therefore the negation of  $r^\Phi(s, a)$ :  $F(s, a) = \gamma\Phi(s', a') - \Phi(s, a) = -r^\Phi(s, a)$ . This leads to the approach of using the negation of  $R^+$  as the immediate reward to train an extra state-action *Value* function  $\Phi$  and the policy simultaneously, with  $r^\Phi(s, a) = -R^+(s, a)$ .  $\Phi$  will be updated by a residual term  $\delta(\Phi)$ :

$$\Phi(s, a) \leftarrow \Phi(s, a) + \beta\delta(\Phi),$$

where  $\delta(\Phi) = -R^+(s, a) + \gamma\Phi(s', a') - \Phi(s, a)$ , and  $\beta$  is the learning rate. Accordingly, the dynamic potential function  $F$  becomes:

$$F_t(s, a) = \gamma\Phi_{t+1}(s', a') - \Phi_t(s, a).$$

The advantage of DPBA is that it provides a framework to allow arbitrary knowledge to be shaped as auxiliary rewards.

Efforts along this line mainly focus on designing different shaping functions  $F(s, a)$ , while little work has addressed the question of *what knowledge can be used to derive this potential function*. One work by [59] proposed to use RS to transfer an expert policy from the source domain ( $M_s$ ) to the target

domain ( $M_t$ ). This approach assumed the existence of two mapping functions,  $M_S$  and  $M_A$ , which can transform the state and action from the source to the target domain. Then the augmented reward is just  $\pi_s((M_S(s), M_A(a)))$ , which is the probability that the mapped state and action will be taken by the expert policy in the source domain. Another work used demonstrated state-action samples from an expert policy to shape rewards [60]. Learning the augmented reward involves a discriminator, which is trained to distinguish samples generated by an expert policy from samples generated by the target policy. The loss of the discriminator is applied to shape rewards to incentivize the learning agent to mimic the expert behavior. This work is a combination of two transfer learning approaches: RS and *Learning from Demonstrations*, the latter of which will be elaborated in Section 4.2.

Besides the single-agent and model-free RL scheme, there have been efforts to apply RS to multi-agent RL [61] and model-based RL [62]. Especially, [61] extended the idea of RS to multi-agent systems, showing that the Nash Equilibria of the underlying stochastic game is unchanged under a potential-based reward shaping structure. [62] applied RS to model-based RL, where the potential function is learned based on the *free space assumption*, an approach to model transition dynamics in the environment.

RS approaches discussed so far are built upon a consensus that the source information for shaping the reward comes *externally*, which coincides with the notion of *knowledge transfer*. Some work of RS also considers the scenario where the augmented reward comes *intrinsically*. *Belief Reward Shaping* was proposed by [63], which utilizes a Bayesian reward shaping framework to generate the potential value that decays with experience, where the potential value comes from the critic itself. Another work is proposed by [63], which aims at better exploiting the environment-provided reward signal by learning potential functions using ideas from graph representation learning. In addition, [64] proposed to learn the potential functions by leveraging graph convolutional networks to perform message passing from rewarding states. The message then can be used as potential functions for reward shaping.

The above RS approaches are summarized in Table 1.

Most RS approaches follow the potential based RS principle that has been developed systematically: from the classical *PTRS* which is built on a *static* potential shaping function of *states*, to *PBA* which generates the potential as a function of both *states* and *actions*, and *DPB* which learns a dynamic potential function of *states* and *time*, to the most recent *DPBA*, which involves a dynamic potential function of *states* and *actions* to be learned as an extra state-action *Value* function in parallel with the environment *Value* function. As an effective transfer learning paradigm, RS has been widely applied to fields including robot training [65], spoken dialogue systems [66], and question answering [67]. It provides a feasible framework for transferring knowledge as the augmented reward and is generally applicable to various RL algorithms. How to integrate RS with other transfer learning approaches to build the potential function, such as *Learning from demonstrations* (Section 4.2) and *Policy Transfer* (Section 4.3) will be an intriguing question for future research.

## 4.2 Learning from Demonstrations

In this section, we review transfer learning techniques in which the transferred knowledge takes the form of external demonstrations. The demonstrations may come from different sources with different qualities: it can be provided by a human expert, a previously learned expert policy, or even a suboptimal policy. For the following discussion, we use  $D_E$  to denote a set of demonstrations, and each element in  $D_E$  is a tuple of transition: *i.e.*  $(s, a, s', r) \in D_E$ . Efforts along this line mostly address a specific transfer learning scenario, *i.e.* the source and the target MDPs are the same:  $\mathcal{M}_s = \mathcal{M}_t$ , although there has been work that learns from demonstrations generated in a different domain [68, 69]. In general, *learning from demonstrations* (*LfD*) is a technique to assist RL by utilizing provided demonstrations for more efficient exploration. Knowledge conveyed in demonstrations encourages agents to explore states which can benefit their policy learning.

Depending on *when* the demonstrations are used for knowledge transfer, approaches can be organized into *offline* methods and *online* methods. For *offline* approaches, demonstrations are either used for pre-training RL components, or used to assist *offline* RL [70, 71]. When leveraging demonstrations for pre-training, RL components such as the value function  $V(s)$  [72], the policy  $\pi$  [73], or even the model of transition dynamics [74], can be initialized by learning from these demonstrations. Specifically, [71] illustrated that using demonstrations for representation learning can remarkably improve various downstream tasks, including both offline and online RL. For the *online* approach, demonstrations are directly used in the online learning stage to guide agent actions for efficient explorations [75]. Most work discussed in this section follows the online transfer paradigm or combines offline pre-training with online RL [76].

Depending on *what* RL frameworks are compatible, work along this line can be categorized into different branches: some adopts the policy-iteration framework [45, 77, 78], others follow a *Q*-learning framework [75, 79], while more recent work follows the policy-gradient framework [32, 60, 76, 80]. Demonstrations have been leveraged in the *policy iterations* framework by [81]. Later, [77] introduced the *Direct Policy Iteration with Demonstrations* (*DPID*) algorithm. This approach

samples complete demonstrated rollouts  $D_E$  from an expert policy  $\pi_E$ , in combination with the self-generated rollouts  $D_\pi$  gathered from the learning agent.  $D_\pi \cup D_E$  are used to learn a Monte-Carlo estimation of the *Q*-value:  $\hat{Q}$ , from which a learning policy can be derived greedily:  $\pi(s) = \arg \max_{a \in \mathcal{A}} \hat{Q}(s, a)$ . This policy  $\pi$  is further regularized by a loss function  $\mathcal{L}(s, \pi_E)$  to minimize its discrepancy from the expert policy decision:  $\mathcal{L}(\pi, \pi_E) = \frac{1}{N_E} \sum_{i=1}^{N_E} \mathbb{1}\{\pi_E(s_i) \neq \pi(s_i)\}$ , where  $N_E$  is the number of expert demonstration samples.

Another work along this line includes the *Approximate Policy Iteration with Demonstration* (*APID*) algorithm, which was proposed by [45] and extended by [78]. Different from *DPID* where both  $D_E$  and  $D_\pi$  are used for value estimation, the *APID* algorithm applies only  $D_\pi$  to approximate on the *Q* function. The expert demonstrations  $D_E$  are used to learn the value function, which, given any state  $s_i$ , renders expert actions  $\pi_E(s_i)$  with higher *Q*-value margins compared with other actions that are not shown in  $D_E$ :

$$Q(s_i, \pi_E(s_i)) - \max_{a \in \mathcal{A} \setminus \pi_E(s_i)} Q(s_i, a) \geq 1 - \xi_i.$$

The term  $\xi_i$  is used to account for the case of imperfect demonstrations. [78] further extended the work of *APID* with a different evaluation loss:  $\mathcal{L}^\pi = \mathbb{E}_{(s, a) \sim D_\pi} \|\mathcal{T}^* Q(s, a) - Q(s, a)\|$ , where  $\mathcal{T}^* Q(s, a) = R(s, a) + \gamma \mathbb{E}_{s' \sim p(\cdot | s, a)} [\max_{a'} Q(s', a')]$ . Their work theoretically converges to the optimal *Q*-function compared with *APID*, as  $\mathcal{L}_\pi$  is minimizing the optimal Bellman residual instead of the empirical norm.

In addition to policy iteration, the following two approaches integrate demonstration data into the TD-learning framework, such as *Q*-learning. Specifically, [75] proposed the *Deep Q-learning from Demonstration* (*DQfD*) algorithm, which maintains two separate replay buffers to store demonstrated data and self-generated data, respectively, so that expert demonstrations can always be sampled with a certain probability. Their work leverages the refined priority replay mechanism [82] where the probability of sampling a transition  $i$  is based on its priority  $p_i$  with a temperature parameter  $\alpha$ :  $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$ . Another work under the *Q*-learning framework was proposed by [79]. Their approach, dubbed as *LfDS*, draws a close connection to the *reward shaping* technique in Section 4.1. It builds the potential function based on a set of expert demonstrations, and the potential value of a given state-action pair is measured by the highest similarity between the given pair and the expert experiences. This augmented reward assigns more credits to state-actions that are more similar to expert demonstrations, which can eventually encourage the agent for expert-like behavior.

Besides *Q*-learning, recent work has integrated *LfD* into the *policy gradient* framework [28, 32, 60, 76, 80]. A representative work along this line is *Generative Adversarial Imitation Learning* (*GAIL*), proposed by [28]. *GAIL* introduced the notion of *occupancy measure*  $d_\pi$ , which is the stationary state-action distributions derived from a policy  $\pi$ . Based on this notion, a new reward function is designed such that maximizing the accumulated new rewards encourages minimizing the distribution divergence between the *occupancy measure* of the current policy  $\pi$  and the expert policy

Methods	MDP difference	Format of shaping reward	Knowledge source
PBRS	$\mathcal{M}_s = \mathcal{M}_t$	$F = \gamma\Phi(s') - \Phi(s)$	X
PBA	$\mathcal{M}_s = \mathcal{M}_t$	$F = \gamma\Phi(s', a') - \Phi(s, a)$	X
DPB	$\mathcal{M}_s = \mathcal{M}_t$	$F = \gamma\Phi(s', t') - \Phi(s, t)$	X
DPBA	$\mathcal{M}_s = \mathcal{M}_t$	$F_t = \gamma\Phi_{t+1}(s', a') - \Phi_t(s, a)$ , $\Phi$ learned as an extra Q function	X
[59]	$\mathcal{S}_s \neq \mathcal{S}_t, \mathcal{A}_s \neq \mathcal{A}_t$	$F_t = \gamma\Phi_{t+1}(s', a') - \Phi_t(s, a)$	$\pi_s$
[60]	$\mathcal{M}_s = \mathcal{M}_t$	$F_t = \gamma\Phi_{t+1}(s', a') - \Phi_t(s, a)$	$D_E$

TABLE 1: A comparison of *reward shaping* approaches. X denotes that the information is not revealed in the paper.

$\pi_E$ . Specifically, the new reward is learned by adversarial training [48]: a discriminator  $D$  is trained to distinguish interactions sampled from the current policy  $\pi$  and the expert policy  $\pi_E$ :

$$J_D = \max_{D: \mathcal{S} \times \mathcal{A} \rightarrow (0,1)} \mathbb{E}_{d_\pi} \log[1 - D(s, a)] + \mathbb{E}_{d_E} \log[D(s, a)]$$

Since  $\pi_E$  is unknown, its state-action distribution  $d_E$  is estimated based on the given expert demonstrations  $D_E$ . It has been proved that, for a optimized discriminator, its output satisfies  $D(s, a) = \frac{d_\pi}{d_\pi + d_E}$ . The output of the discriminator is used as new rewards to encourage distribution matching, with  $r'(s, a) = -\log(1 - D(s, a))$ . The RL process is naturally altered to perform distribution matching by optimizing the following minimax objective:

$$\max_{\pi} \min_D J(\pi, D) := \mathbb{E}_{d_\pi} \log[1 - D(s, a)] + \mathbb{E}_{d_E} \log[D(s, a)].$$

Although *GAIL* is more related to *imitation learning* than *LfD*, its philosophy of using expert demonstrations for distribution matching has inspired other *LfD* algorithms. For example, [80] extended *GAIL* with an algorithm called *Policy Optimization from Demonstrations (POfD)*, which combines the discriminator reward with the environment reward, so that the agent is trained to maximize the accumulated environment rewards (RL objective) as well as performing distribution matching (imitation learning objective):

$$\max_{\theta} = \mathbb{E}_{d_\pi} [r(s, a)] - \lambda D_{JS}[\pi || d_E]. \quad (4)$$

Both *GAIL* and *POfD* are under an *on-policy* RL framework. To further improve the sample efficiency of transfer learning, some *off-policy* algorithms have been proposed, such as *DDPGfD* [60] which is built upon the *DDPG* framework. *DDPGfD* shares a similar idea as *DQfD* in that they both use a second replay buffer for storing demonstrated data, and each demonstrated sample holds a sampling priority  $p_i$ . For a demonstrated sample, its priority  $p_i$  is augmented with a constant bias  $\epsilon_D > 0$  for encouraging more frequent sampling of expert demonstrations:

$$p_i = \delta_i^2 + \lambda \|\nabla_a Q(s_i, a_i | \theta^Q)\|^2 + \epsilon + \epsilon_D,$$

where  $\delta_i$  is the TD-residual for transition,  $\|\nabla_a Q(s_i, a_i | \theta^Q)\|^2$  is the loss applied to the actor, and  $\epsilon$  is a small positive constant to ensure all transitions are sampled with some probability. Another work also adopted the *DDPG* framework to learn from demonstrations [76]. Their approach differs from *DDPGfD* in that its objective function is augmented with a *Behavior Cloning Loss* to encourage imitating on provided demonstrations:  $\mathcal{L}_{BC} = \sum_{i=1}^{|D_E|} \|\pi(s_i | \theta_\pi) - a_i\|^2$ .

To further address the issue of *suboptimal demonstrations*, in [76] the form of *Behavior Cloning Loss* is altered based on

the critic output, so that only demonstration actions with higher  $Q$  values will lead to the loss penalty:

$$\mathcal{L}_{BC} = \sum_{i=1}^{|D_E|} \|\pi(s_i | \theta_\pi) - a_i\|^2 \mathbb{1}[Q(s_i, a_i) > Q(s_i, \pi(s_i))].$$

There are several challenges faced by *LfD*, one of which is the *imperfect demonstrations*. Previous approaches usually presume near-oracle demonstrations. However, demonstrations can also be biased estimations of the environment or even from a sub-optimal policy [32]. Current solutions to imperfect demonstrations include altering the objective function. For example, [45] leveraged the hinge-loss function to allow occasional violations of the property that  $Q(s_i, \pi_E(s_i)) - \max_{a \in \mathcal{A} \setminus \pi_E(s_i)} Q(s_i, a) \geq 1$ . Some other work uses regularizations on the objective to alleviate overfitting on biased data [75, 82]. A different strategy to confront the sub-optimality is to leverage those sub-optimal demonstrations only to boost the initial learning stage. Specifically, in the same spirit of *GAIL*, [32] proposed *Self-Adaptive Imitation Learning (SAIL)*, which learns from sub-optimal demonstrations using generative adversarial training while gradually selecting self-generated trajectories with high qualities to replace less superior demonstrations.

Another challenge faced by *LfD* is *covariate drift* ([83]): demonstrations may be provided in limited numbers, which results in the learning agent lacking guidance on states that are unseen in the demonstration dataset. This challenge is aggravated in MDPs with sparse reward feedbacks, as the learning agent cannot obtain much supervision information from the environment either. Current efforts to address this challenge include encouraging explorations by using an entropy-regularized objective [52], decaying the effects of demonstration guidance by softening its regularization on policy learning over time [31], and introducing *disagreement regularizations* by training an ensemble of policies based on the given demonstrations, where the variance among policies serves as a cost (negative reward) function [84].

We summarize the above-discussed approaches in Table 2. In general, demonstration data can help in both *offline* pre-training for better initialization and *online* RL for efficient exploration. During the RL phase, demonstration data can be used together with self-generated data to encourage expert-like behaviors (*DDPGfD*, *DQfD*), to shape value functions (*APID*), or to guide the policy update in the form of an auxiliary objective function (*PID*, *GAIL*, *POfD*). The current RL framework used for *LfD* includes policy iteration, *Q*-learning, and policy gradient. Developing more general *LfD* approaches that are agnostic to RL frameworks and can learn from sub-optimal or limited demonstrations would be the next focus for this research domain.

Methods	Optimality guarantee	Format of transferred demonstrations	Reinforcement learning framework
DQfD	✗	Cached transitions in the replay buffer	DQN
LfDS	✗	Reward shaping function	DQN
GAIL	✓	Reward shaping function: $-\lambda \log(1 - D(s, a))$	TRPO
POfD	✓	Reward shaping function: $r(s, a) - \lambda \log(1 - D(s, a))$	TRPO, PPO
DDPGfD	✓	Increasing sampling priority	DDPG
[76]	✓	Increasing sampling priority and behavior cloning loss	DDPG
DPID	✓	Indicator binary-loss : $\mathcal{L}(s_i) = \mathbb{1}\{\pi_E(s_i) \neq \pi(s_i)\}$	API
APID	✗	Hinge loss on the marginal-loss: $[\mathcal{L}(Q, \pi, \pi_E)]_+$	API
APID extend	✓	Marginal-loss: $\mathcal{L}(Q, \pi, \pi_E)$	API
SAIL	✗	Reward shaping function: $r(s, a) - \lambda \log(1 - D(s, a))$	DDPG

TABLE 2: A comparison of *learning from demonstration* approaches.

### 4.3 Policy Transfer

In this section, we review a transfer learning approach dubbed as *policy transfer*, where the external knowledge takes the form of pretrained policies from one or multiple source domains. Work discussed in this section is built upon a *many-to-one* problem setting, which we describe as below:

**Policy Transfer.** A set of teacher policies  $\pi_{E_1}, \pi_{E_2}, \dots, \pi_{E_K}$  are trained on a set of source domains  $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_K$ , respectively. A student policy  $\pi$  is learned for a target domain by leveraging knowledge from  $\{\pi_{E_i}\}_{i=1}^K$ .

For the *one-to-one* scenario, which contains only one teacher policy, one can consider it as a special case of the above problem setting with  $K = 1$ . Next, we categorize recent work of policy transfer into two techniques: policy distillation and policy reuse.

#### 4.3.1 Transfer Learning via Policy Distillation

The term *knowledge distillation* was proposed by [85] as an approach of knowledge ensemble from multiple teacher models into a single student model. This technique is later extended from the field of supervised learning to RL. Since the student model is usually shallower than the teacher model and can perform across multiple teacher tasks, policy distillation is also considered as an effective approach of model compression [86].

The idea of knowledge distillation has been recently applied to the field of RL to enable *policy distillation*. Conventional policy distillation approaches transfer the teacher policy in a supervised learning paradigm [87, 88]. Specifically, a student policy is learned by minimizing the divergence of action distributions between the teacher policy  $\pi_E$  and student policy  $\pi_\theta$ , which is denoted as  $\mathcal{H}^\times(\pi_E(\tau_t)|\pi_\theta(\tau_t))$ :

$$\min_{\theta} \mathbb{E}_{\tau \sim \pi_E} \left[ \sum_{t=1}^{|\tau|} \nabla_{\theta} \mathcal{H}^\times(\pi_E(\tau_t)|\pi_\theta(\tau_t)) \right].$$

The above expectation is taken over trajectories sampled from the teacher policy  $\pi_E$ , which therefore makes this approach *teacher distillation*. A representative example of work along this line is [87], in which  $N$  teacher policies are learned for  $N$  source tasks separately, and each teacher yields a dataset  $D^E = \{s_i, q_i\}_{i=0}^N$  consisting of observations (states)  $s$  and vectors of the corresponding  $Q$ -values  $q$ , such that  $q_i = [Q(s_i, a_1), Q(s_i, a_2), \dots | a_j \in \mathcal{A}]$ . Teacher policies are further distilled to a single student agent  $\pi_\theta$  by minimizing the KL-Divergence between each teacher policy  $\pi_{E_i}(a|s)$  and

the student policy  $\pi_\theta$ , approximated using the dataset  $D^E$ :  $\min_{\theta} \mathcal{D}_{KL}(\pi^E|\pi_\theta) \approx \sum_{i=1}^{|D^E|} \text{softmax}\left(\frac{q_i^E}{\tau}\right) \ln\left(\frac{\text{softmax}(q_i^E)}{\text{softmax}(q_i^{\theta})}\right)$ .

Another policy distillation approach is *student distillation* [46, 89], which is similar to teacher distillation, except that during the optimization step, the expectation is taken over trajectories sampled from the student policy instead of the teacher policy:  $\min_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=1}^{|\tau|} \nabla_{\theta} \mathcal{H}^\times(\pi_E(\tau_t)|\pi_\theta(\tau_t)) \right]$ .

[46] provides a nice summarization of the related work on both kinds of distillation approaches. While it is feasible to combine both distillation approaches [83], we observe that more recent work focuses on student distillation, which empirically shows better exploration ability compared to teacher distillation, especially when the teacher policy is deterministic.

Taking an alternative perspective, there are two approaches of distilling the knowledge from teacher policies to a student: (1) minimizing the cross-entropy between the teacher and student policy distributions over actions [89, 90]; and (2) maximizing the probability that the teacher policy will visit trajectories generated by the student, i.e.  $\max_{\theta} P(\tau \sim \pi_E|\tau \sim \pi_\theta)$  [91, 92]. One example of approach (1) is the *Actor-mimic* algorithm [89]. This algorithm distills the knowledge of expert agents into the student by minimizing the cross entropy between the student policy  $\pi_\theta$  and each teacher policy  $\pi_{E_i}$  over actions:  $\mathcal{L}^i(\theta) = \sum_{a \in \mathcal{A}_{E_i}} \pi_{E_i}(a|s) \log_{\pi_\theta}(a|s)$ , where each teacher agent is learned using a DQN framework, whose policy is therefore derived from the Boltzmann distributions over the  $Q$ -function output:  $\pi_{E_i}(a|s) = \frac{e^{\tau^{-1} Q_{E_i}(s, a)}}{\sum_{a' \in \mathcal{A}_{E_i}} e^{\tau^{-1} Q_{E_i}(s, a')}}$ . An instantiation of approach (2) is the *Distral* algorithm [91], which learns a *centroid* policy  $\pi_\theta$  that is derived from  $K$  teacher policies. The knowledge in each teacher  $\pi_{E_i}$  is distilled to the centroid and get transferred to student policies, while both the transition dynamics  $\mathcal{T}_i$  and reward distributions  $\mathcal{R}_i$  for source domain  $\mathcal{M}_i$  are heterogeneous. Specifically, the distilled policy (student) is learned to perform in different domains by maximizing a multi-task learning objective  $\max_{\theta} \sum_{i=1}^K J(\pi_\theta, \pi_{E_i})$ , where

$$J(\pi_\theta, \pi_{E_i}) = \sum_t \mathbb{E}_{(s_t, a_t) \sim \pi_\theta} \left[ \sum_{t \geq 0} \gamma^t (r_i(a_t, s_t) + \frac{\alpha}{\beta} \log \pi_\theta(a_t|s_t) - \frac{1}{\beta} \log(\pi_{E_i}(a_t|s_t))) \right],$$

in which both  $\log \pi_\theta(a_t|s_t)$  and  $\pi_\theta$  are used as augmented

rewards. Therefore, the above approach also draws a close connection to *Reward Shaping* (Section 4.1). In effect, the  $\log(\pi_\theta(a_t|s_t))$  term guides the learning policy  $\pi_\theta$  to yield actions that are more likely to be generated by the teacher policy, whereas the entropy term  $-\log(\pi_{E_i}(a_t|s_t))$  encourages exploration. A similar approach was proposed by [90] which only uses the cross-entropy between teacher and student policy  $\lambda\mathcal{H}(\pi_E(a_t|s_t)||\pi_\theta(a_t|s_t))$  to reshape rewards. Moreover, they adopted a dynamically fading coefficient to alleviate the effect of the augmented reward so that the student policy becomes independent of the teachers after certain optimization iterations.

#### 4.3.2 Transfer Learning via Policy Reuse

Another policy transfer approach is *policy reuse*, which directly reuses policies from source tasks to build the target policy. The notion of policy reuse was proposed by [93], which directly learns the target policy as a weighted combination of different source-domain policies, while the probability for each source domain policy to be used is related to its expected performance gain in the target domain:  $P(\pi_{E_i}) = \frac{\exp(tW_i)}{\sum_{j=0}^K \exp(tW_j)}$ , where  $t$  is a dynamic temperature parameter that increases over time. Under a  $Q$ -learning framework, the  $Q$ -function of the target policy is learned in an iterative scheme: during every learning episode,  $W_i$  is evaluated for each expert policy  $\pi_{E_i}$ , and  $W_0$  is obtained for the learning policy, from which a reuse probability  $P$  is derived. Next, a behavior policy is sampled from this probability  $P$ . After each training episode, both  $W_i$  and the temperature  $t$  for calculating the reuse probability is updated accordingly. One limitation of this approach is that the  $W_i$ , i.e. the expected return of each expert policy on the target task, needs to be evaluated frequently. This work was implemented in a tabular case, leaving the scalability issue unresolved. More recent work by [94] extended the *policy improvement* theorem [95] from one to multiple policies, which is named as *Generalized Policy Improvement*. We refer its main theorem as follows:

**Theorem.** [Generalized Policy Improvement (GPI)] Let  $\{\pi_i\}_{i=1}^n$  be  $n$  policies and let  $\{\hat{Q}^{\pi_i}\}_{i=1}^n$  be their approximated action-value functions, s.t:  $|Q^{\pi_i}(s, a) - \hat{Q}^{\pi_i}(s, a)| \leq \epsilon \forall s \in \mathcal{S}, a \in \mathcal{A}$ , and  $i \in [n]$ . Define  $\pi(s) = \arg \max_a \max_i \hat{Q}^{\pi_i}(s, a)$ , then:  $Q^\pi(s, a) \geq \max_i Q^{\pi_i}(s, a) - \frac{2}{1-\gamma}\epsilon, \forall s \in \mathcal{S}, a \in \mathcal{A}$ .

Based on this theorem, a policy improvement approach can be naturally derived by greedily choosing the action which renders the highest  $Q$ -value among all policies for a given state. Another work along this line is [94], in which an expert policy  $\pi_{E_i}$  is also trained on a different source domain  $\mathcal{M}_i$  with reward function  $\mathcal{R}_i$ , so that  $Q_{\mathcal{M}_0}^\pi(s, a) \neq Q_{\mathcal{M}_i}^\pi(s, a)$ . To efficiently evaluate the  $Q$ -functions of different source policies in the target MDP, a disentangled representation  $\psi(s, a)$  over the states and actions is learned using neural networks and is generalized across multiple tasks. Next, a task (reward) mapper  $\mathbf{w}_i$  is learned, based on which the  $Q$ -function can be derived:

$$Q_i^\pi(s, a) = \psi(s, a)^T \mathbf{w}_i.$$

[94] proved that the loss of GPI is bounded by the difference between the source and the target tasks. In addition to

policy-reuse, their approach involves learning a shared representation  $\psi(s, a)$ , which is also a form of transferred knowledge and will be elaborated more in Section 4.5.2.

We summarize the abovementioned policy transfer approaches in Table 3. In general, policy transfer can be realized by *knowledge distillation*, which can be either optimized from the student's perspective (*student distillation*), or from the teacher's perspective (*teacher distillation*). Alternatively, teacher policies can also be directly *reused* to update the target policy. All approaches discussed so far presumed one or multiple *expert* policies, which are always at the disposal of the learning agent. Questions such as *How to leverage imperfect policies for knowledge transfer*, and *How to refer to teacher policies within a budget*, are still open to be resolved by future research along this line.

#### 4.4 Inter-Task Mapping

In this section, we review transfer learning approaches that utilize *mapping functions* between the source and the target domains to assist knowledge transfer. Research in this domain can be analyzed from two perspectives: (1) *which domain does the mapping function apply to*, and (2) *how is the mapped representation utilized*. Most work discussed in this section shares a common assumption as below:

**Assumption.** [Existence of Domain Mapping] *One-to-one mappings exist between the source domain  $\mathcal{M}_s$  and the target domain  $\mathcal{M}_t$ .*

Earlier work along this line requires a *given mapping function* [53, 96]. One examples is [53] which assumes that each target state (action) has a unique correspondence in the source domain, and two mapping functions  $X_S, X_A$  are provided over the state space and the action space, respectively, so that  $X_S(\mathcal{S}^t) \rightarrow \mathcal{S}^s, X_A(\mathcal{A}^t) \rightarrow \mathcal{A}^s$ . Based on  $X_S$  and  $X_A$ , a mapping function over the  $Q$ -values  $M(Q_s) \rightarrow Q_t$  can be derived accordingly. Another work is done by [96] which transfers *advice* as the knowledge between two domains. In their settings, the *advice* comes from a human expert who provides the mapping function over the  $Q$ -values in the source domain and transfers it to the learning policy for the target domain. This advice encourages the learning agent to prefer certain good actions over others, which equivalently provides a relative ranking of actions in the new task.

More later research tackles the inter-task mapping problem by *automatically* learning a mapping function [97–99]. Most work learns a mapping function over the *state* space or a subset of the state space. In their work, state representations are usually divided into *agent-specific* and *task-specific* representations, denoted as  $s_{agent}$  and  $s_{env}$ , respectively. In [97] and [98], the mapping function is learned on the *agent-specific* sub state, and the mapped representation is applied to reshape the immediate reward. For [97], the invariant feature space mapped from  $s_{agent}$  can be applied across agents who have distinct action space but share some morphological similarity. Specifically, they assume that both agents have been trained on the same *proxy* task, based on which the mapping function is learned. The mapping function is learned using an encoder-decoder neural network structure [100] in order to reserve as much information about the source domain as possible. While transferring knowledge

Citation	Transfer approach	MDP difference	Reinforcement learning framework	Metrics
[87]	Distillation	$\mathcal{S}, \mathcal{A}$	DQN	$ap$
[88]	Distillation	$\mathcal{S}, \mathcal{A}$	DQN	$ap, ps$
[89]	Distillation	$\mathcal{S}, \mathcal{A}$	Soft Q-learning	$ap, ar, ps$
[91]	Distillation	$\mathcal{S}, \mathcal{A}$	A3C	$ap, pe, tt$
[93]	Reuse	$\mathcal{R}$	Tabular Q-learning	$ap$
[94]	Reuse	$\mathcal{R}$	DQN	$ap, ar$

TABLE 3: A comparison of *policy transfer* approaches.

from the source agent to the target agent on a new task, the environment reward is augmented with a shaped reward term to encourage the target agent to imitate the source agent on the embedded feature space:

$$r'(s, \cdot) = \alpha \|f(s_{agent}^s; \theta_f) - g(s_{agent}^t; \theta_g)\|,$$

where  $f(s_{agent}^s)$  is the agent-specific state in the source domain, and  $g(s_{agent}^t)$  is for the target domain.

[99] applied the *Unsupervised Manifold Alignment (UMA)* approach [101] to automatically learn the state mapping between tasks. In their approach, trajectories are collected from both the source and the target domain to learn a mapping between states. While applying policy gradient learning, trajectories from  $\mathcal{M}_t$  are first mapped back to the source:  $\tau_t \rightarrow \tau_s$ , then an expert policy in the source domain is applied to each initial state of those trajectories to generate near-optimal trajectories  $\tilde{\tau}_s$ , which are further mapped to the target domain:  $\tilde{\tau}_s \rightarrow \tilde{\tau}_t$ . The deviation between  $\tilde{\tau}_t$  and  $\tau_t$  are used as a loss to be minimized in order to improve the target policy. Similar ideas of using *UMA* to assist transfer by inter-task mapping can also be found in [102] and [103].

In addition to approaches that utilizes mapping over states or actions, [104] proposed to learn an inter-task mapping over the *transition dynamics* space:  $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ . Their work assumes that the source and target domains are different in terms of the transition space dimensionality. Transitions from both the source domain  $\langle s^s, a^s, s'^s \rangle$  and the target domain  $\langle s^t, a^t, s'^t \rangle$  are mapped to a latent space  $Z$ . Given the latent feature representations, a similarity measure can be applied to find a correspondence between the source and target task triplets. Triplet pairs with the highest similarity in this feature space  $Z$  are used to learn a mapping function  $\mathcal{X}: \langle s^t, a^t, s'^t \rangle = \mathcal{X}(\langle s^s, a^s, s'^s \rangle)$ . After the transition mapping, states sampled from the expert policy in the source domain can be leveraged to render beneficial states in the target domain, which assists the target agent learning with a better initialization performance. A similar idea of mapping transition dynamics can be found in [105], which, however, requires a stronger assumption on the similarity of the transition probability and the state representations between the source and the target domains.

As summarized in Table 4, for transfer learning approaches that utilize an inter-task mapping, the mapped knowledge can be (a subset of) the state space [97, 98], the *Q*-function [53], or (representations of) the state-action-sate transitions [104]. In addition to being directly applicable in the target domain [104], the mapped representation can also be used as an augmented shaping reward [97, 98] or a loss objective [99] in order to guide the agent learning in the target domain.

## 4.5 Representation Transfer

In this section, we review approaches the transfer knowledge are feature representations, such as representations learned for the value function or *Q*-function. Approaches discussed in this section are developed based on the powerful approximation ability of deep neural networks and are built upon the following consensual assumption:

**Assumption.** [Existence of Task-Invariance Subspace]

*The state space ( $\mathcal{S}$ ), action space ( $\mathcal{A}$ ), or even reward space ( $\mathcal{R}$ ) can be disentangled into orthogonal subspaces, some of which are task-invariant and are shared by both the source and target domains, such that knowledge can be transferred between domains on the universal sub-space.*

We organize recent work along this line into two subtopics: i) approaches that directly reuse representations from the source domain (Section 4.5.1), and ii) approaches that learn to disentangle the source domain representations into independent sub-feature representations, some of which are on the universal feature space shared by both the source and the target domains (Section 4.5.2).

### 4.5.1 Reusing Representations

A representative work of reusing representations is [106], which proposed the *progressive neural network* structure to enable knowledge transfer across multiple RL tasks in a progressive way. A progressive network is composed of multiple *columns*, where each column is a policy network for training one specific task. It starts with one single column for training the first task, and then the number of columns increases with the number of new tasks. While training on a new task, neuron weights on the previous columns are frozen, and representations from those frozen tasks are applied to the new column via a collateral connection to assist in learning the new task. This process can be mathematically generalized as follows:  $h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j < k} U_i^{(k:j)} h_{i-1}^{(j)}\right)$ , where  $h_i^{(k)}$  is the  $i$ -th hidden layer for task (column)  $k$ ,  $W_i^{(k)}$  is the associated weight matrix, and  $U_i^{(k:j)}$  are the lateral connections from layer  $i-1$  of previous tasks to the current layer of task  $k$ .

Although *progressive network* is an effective multi-task approach, it comes with a cost of giant network structure, as the network grows proportionally with the number of incoming tasks. A later framework called *PathNet* is proposed by [107] which alleviates this issue by using a network with a fixed size. *PathNet* contains *pathways*, which are subsets of neurons whose weights contain the knowledge of previous tasks and are frozen during training on new tasks. The population of *pathway* is evolved using a tournament selection genetic algorithm [108].

Citation	Algorithm	MDP difference	Mapping function	Usage of mapping
[53]	SARSA	$\mathcal{S}_t \neq \mathcal{S}_s, \mathcal{A}_s \neq \mathcal{A}_t$	$M(Q_s) \rightarrow Q_t$	$Q$ value reuse
[96]	<i>Q</i> -learning	$\mathcal{A}_s \neq \mathcal{A}_t, \mathcal{R}_s \neq \mathcal{R}_t$	$M(Q_s) \rightarrow \text{advice}$	Relative $Q$ ranking
[97]	Generally Applicable	$\mathcal{S}_s \neq \mathcal{S}_t$	$M(s_t) \rightarrow r'$	Reward shaping
[98]	$SARSA(\lambda)$	$\mathcal{S}_s \neq \mathcal{S}_t, \mathcal{R}_s \neq \mathcal{R}_t$	$M(s_t) \rightarrow r'$	Reward shaping
[99]	Fitted Value Iteration	$\mathcal{S}_s \neq \mathcal{S}_t$	$M(s_s) \rightarrow s_t$	Penalty loss on state deviation from expert policy
[105]	Fitted $Q$ Iteration	$\mathcal{S}_s \times \mathcal{A}_s \neq \mathcal{S}_t \times \mathcal{A}_t$	$M((s_s, a_s, s'_s) \rightarrow (s_t, a_t, s'_t))$	Reduce random exploration
[104]	No constraint	$\mathcal{S}_s \times \mathcal{A}_s \neq \mathcal{S}_t \times \mathcal{A}_t$	$M((s_s, a_s, s'_s) \rightarrow (s_t, a_t, s'_t))$	Reduce random exploration

TABLE 4: A comparison of *inter-task mapping* approaches.

Another approach of reusing representations for transfer learning is modular networks [109–111]. For example, [109] proposed to decompose the policy network into a task-specific module and agent-specific module. Specifically, let  $\pi$  be a policy performed by any agent (robot)  $r$  over the task  $\mathcal{M}_k$  as a function  $\phi$  over states  $s$ , it can be decomposed into two sub-modules  $g_k$  and  $f_r$ , *i.e.*:

$$\pi(s) := \phi(s_{env}, s_{agent}) = f_r(g_k(s_{env}), s_{agent}),$$

where  $f_r$  is the agent-specific module and  $g_k$  is the task-specific module. Their core idea is that the task-specific module can be applied to different agents performing the same task, which serves as a transferred knowledge. Accordingly, the agent-specific module can be applied to different tasks for the same agent.

A model-based approach along this line is [111], which learns a model to map the state observation  $s$  to a latent-representation  $z$ . The transition probability is modeled on the latent space instead of the original state space, *i.e.*  $\hat{z}_{t+1} = f_\theta(z_t, a_t)$ , where  $\theta$  is the parameter of the transition model,  $z_t$  is the latent-representation of the state observation, and  $a_t$  is the action accompanying that state. Next, a *reward* module learns the value function as well as the policy from the latent space  $z$  using an actor-critic framework. One potential benefit of this latent representation is that knowledge can be transferred across tasks that have different rewards but share the same transition dynamics, in which case the *dynamics* module can be directly applied to the target domain.

#### 4.5.2 Disentangling Representations

Methods discussed in this section mostly focus on learning a *disentangled* representation. Specifically, we elaborate on transfer learning approaches that are derived from two techniques: *Successor Representation* (*SR*) and *Universal Value Function Approximating* (*UVFA*).

**Successor Representations (SR)** is an approach to decouple the state features of a domain from its reward distributions. It enables knowledge transfer across multiple domains:  $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_K\}$ , so long as the only difference among them is the reward distributions:  $\mathcal{R}_i \neq \mathcal{R}_j$ . SR was originally derived from neuroscience, until [112] proposed to leverage it as a generalization mechanism for state representations in the RL domain.

Different from the  $v$ -value or  $Q$ -value that describes states as dependent on the reward distribution of the MDP, SR features a state based on the *occupancy measure* of its successor states. Specifically, SR decomposes the value function of any policy into two independent components,  $\psi$  and  $R$ :  $V^\pi(s) = \sum_{s'} \psi(s, s') \mathbf{w}(s')$ , where  $\mathbf{w}(s')$  is a

reward mapping function that maps states to scalar rewards, and  $\psi$  is the *SR* which describes any state  $s$  as the occupancy measure of the future occurred states when following  $\pi$ :  $\psi(s, s') = \mathbb{E}_\pi [\sum_{i=t}^{\infty} \gamma^{i-t} \mathbb{1}[S_i = s'] | S_t = s]$ , with  $\mathbb{1}[S = s'] = 1$  as an indicator function.

The *successor* nature of *SR* makes it learnable using any TD-learning algorithms. Especially, [112] proved the feasibility of learning such representation in a tabular case, in which the state transitions can be described using a matrix. SR was later extended by [94] from three perspectives: (i) the feature domain of SR is extended from states to state-action pairs; (ii) deep neural networks are used as function approximators to represent the SR  $\psi^\pi(s, a)$  and the *reward mapper*  $\mathbf{w}$ ; (iii) Generalized policy improvement (GPI) algorithm is introduced to accelerate policy transfer for multi-tasks facilitated by the SR framework (See Section 4.3.2 for more details about GPI). These extensions, however, are built upon a stronger assumption about the MDP.

**Assumption.** [Linearity of Reward Distributions] *The reward functions of all tasks can be computed as a linear combination of a fixed set of features:  $r(s, a, s') = \phi(s, a, s')^\top \mathbf{w}$ , where  $\phi(s, a, s') \in \mathbb{R}^d$  denotes the latent representation of the state transition, and  $\mathbf{w} \in \mathbb{R}^d$  is the task-specific reward mapper.*

Based on this assumption, SR can be decoupled from the rewards when evaluating the  $Q$ -function of any policy  $\pi$  in a task  $\mathcal{M}_i$  with a reward function  $\mathcal{R}_i$ :

$$\begin{aligned} Q_i^\pi(s, a) &= \mathbb{E}^\pi[\phi_{t+1}^\top \mathbf{w}_i + \gamma \phi_{t+2}^\top \mathbf{w}_i + \dots | S_t = s, A_t = a] \\ &= \psi^\pi(s, a)^\top \mathbf{w}_i. \end{aligned} \quad (5)$$

The advantage of SR is that, when the knowledge of  $\psi^\pi(s, a)$  in  $\mathcal{M}_s$  is observed, one can quickly get the performance evaluation of the same policy in  $\mathcal{M}_t$  by replacing  $\mathbf{w}_s$  with  $\mathbf{w}_t$ :  $Q_{\mathcal{M}_t}^\pi = \psi^\pi(s, a) \mathbf{w}_t$ . Similar ideas of learning SR as a TD-algorithm on a latent representation  $\phi(s, a, s')$  can also be found in [113, 114]. Specifically, the work of [113] was developed based on a weaker assumption about the reward function: Instead of requiring linearly-decoupled rewards, the latent space  $\phi(s, a, s')$  is learned in an encoder-decoder structure to ensure that the information loss is minimized when mapping states to the latent space. This structure, therefore, comes with an extra cost of learning a decoder  $f_d$  to reconstruct the state:  $f_d(\phi(s_t)) \approx s_t$ .

An intriguing question faced by the SR approach is: *Is there a way that evades the linearity assumption about reward functions and still enables learning the SR without extra modular cost?* An extended work of SR [54] answered this question affirmatively, which proved that the reward functions does not necessarily have to follow the linear structure, yet at the

cost of a looser performance lower-bound while applying the GPI approach for policy improvement. Especially, rather than learning a reward-agnostic latent feature  $\phi(s, a, s') \in \mathbb{R}^d$  for multiple tasks, [54] aims to learn a matrix  $\phi(s, a, s') \in \mathbb{R}^{D \times d}$  to interpret the basis functions of the latent space instead, where  $D$  is the number of seen tasks. Assuming  $k$  out of  $D$  tasks are linearly independent, this matrix forms  $k$  basis functions for the latent space. Therefore, for any unseen task  $\mathcal{M}_i$ , its latent features can be built as a linear combination of these basis functions, as well as its reward functions  $r_i(s, a, s')$ . Based on the idea of basis-functions for a task's latent space, they proposed that  $\phi(s, a, s')$  can be approximated as learning  $\mathbb{R}(s, a, s')$  directly, where  $\mathbb{R}(s, a, s') \in \mathbb{R}^D$  is a vector of reward functions for each seen task:

$$\mathbb{R}(s, a, s') = [r_1(s, a, s'); r_2(s, a, s'), \dots, r_D(s, a, s')].$$

Accordingly, learning  $\psi(s, a)$  for any policy  $\pi_i$  in  $\mathcal{M}_i$  becomes equivalent to learning a collection of Q-functions:

$$\overset{\sim}{\psi}^{\pi_i}(s, a) = [Q_1^{\pi_i}(s, a), Q_2^{\pi_i}(s, a), \dots, Q_D^{\pi_i}(s, a)].$$

A similar idea of using reward functions as features to represent unseen tasks is also proposed by [115], which, however, assumes the  $\psi$  and  $w$  as observable quantities from the environment.

**Universal Function Approximation (UVFA)** is an alternative approach of learning disentangled state representations [49]. Same as SR, UVFA allows transfer learning for multiple tasks which differ only by their reward functions (goals). Different from SR which focuses on learning a reward-agnostic state representation, UVFA aims to find a function approximator that is generalized for both states and goals. The UVFA framework is built on a specific problem setting:

**Goal Conditional RL:** Task goals are defined in terms of states, e.g. given the state space  $\mathcal{S}$  and the goal space  $\mathcal{G}$ , it satisfies that  $\mathcal{G} \subseteq \mathcal{S}$ .

One instantiation of this problem setting can be an agent exploring different locations in a maze, where the goals are described as certain locations inside the maze. Under this problem setting, a UVFA module can be decoupled into a state embedding  $\phi(s)$  and a goal embedding  $\psi(g)$ , by applying the technique of matrix factorization to a reward matrix describing the goal-conditional task.

One merit of UVFA resides in its transferrable embedding  $\phi(s)$  across tasks which only differ by goals. Another benefit is its ability of continual learning when the set of goals keeps expanding over time. On the other hand, a key challenge of UVFA is that applying the matrix factorization is time-consuming, which makes it a practical concern for complex environments with large state space  $|\mathcal{S}|$ . Even with the learned embedding networks, the third stage of fine-tuning these networks via end-to-end training is still necessary. Authors of the paper refer to the *OptSpac* tool for matrix factorization [116].

UVFA has been connected to SR by [54], in which a set of independent rewards (tasks) themselves can be used as features for state representations. Another extended work that combines UVFA with SR is called *Universal Successor Feature Approximator (USFA)*, which is proposed by [117]. Following the same linearity assumption, USFA is proposed

as a function over a triplet of the state, action, and a policy embedding  $z: \phi(s, a, z) : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^k \rightarrow \mathbb{R}^d$ , where  $z$  is the output of a *policy-encoding mapping*  $z = e(\pi) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^k$ . Based on USFA, the  $Q$ -function of any policy  $\pi$  for a task specified by  $w$  can be formalized as the product of a reward-agnostic *Universal Successor Feature (USF)*  $\psi$  and a reward mapper  $w$ :  $Q(s, a, w, z) = \psi(s, a, z)^\top w$ . The  $Q$ -function representation is distinct from Equation (5), as the  $\psi(s, a, z)$  is generalized over multiple policies, with each denoted by  $z$ . Facilitated by the disentangled rewards and policy generalization, [117] further introduced a generalized TD-error as a function over tasks  $w$  and policy  $z$ , which allows them to approximate the  $Q$ -function of any policy on any task using a TD-algorithm.

#### 4.5.3 Discussion

We provide a summary of the discussed work in this section in Table 5. In general, representation transfer approaches can facilitate transfer learning in many ways, with certain shared assumptions about some task-invariant property. Most of them assume that tasks are different only in terms of their reward distributions while sharing the same states (or actions or transitions) probabilities. Other stronger assumptions include (i) decoupled dynamics, rewards [94], or policies [117] from the  $Q$ -function representations, and (ii) the feasibility of defining tasks in terms of states [117]. Based on those assumptions, approaches such as TD-algorithms [54] or matrix-factorization [49] become applicable to learn such disentangled representations. To further exploit the effectiveness of disentangled structure, we think that generalization approaches, which allow changing dynamics or state distributions, are important future work that is worth more attention in this domain.

As an intriguing research topic, there are unresolved questions along the line of representation transfer. One is *how to handle drastic changes of reward functions between domains*. As discussed in [118], good policies in one MDP may perform poorly in another due to the fact that beneficial states or actions in  $\mathcal{M}_s$  may become detrimental in  $\mathcal{M}_t$  with totally different reward functions. Learning a set of basis functions [54] to represent unseen tasks (reward functions), or decoupling policies from  $Q$ -function representation [117] may serve as a good start to address this issue, as they propose a generalized latent space, from which different tasks (reward functions) can be interpreted. However, the limitation of this work is that it is not clear how many and what kind of sub-tasks need to be learned to make the latent space generalizable enough to interpret unseen tasks.

Another question is *how to generalize the representation framework to allow transfer learning across domains with different dynamics (or state-action spaces)*. A learned SR might not be transferrable to an MDP with different transition dynamics, as the distribution of occupancy measure for successor states may no longer hold. Potential solutions may include model-based approaches that approximate the dynamics directly or training a latent representation space for states using multiple tasks with different dynamics for better generalization [119]. Alternatively, transfer learning mechanisms from the supervised learning domain, such as meta-learning, which enables the ability of fast adaptation to new tasks [120], or importance sampling [121], which can

compensate for the prior distribution changes [10], might also shed light on this question.

## 5 APPLICATIONS

In this section, we summarize recent applications of RL based transfer learning techniques.

**Robotics learning** is a practical domain for RL applications. [122] provided a comprehensive summary of RL in robotics learning. A classical approach along this direction is *robotics learning from demonstrations*, where expert demonstrations from humans or other robots are leveraged as the knowledge source. [123] provided a comprehensive survey along this direction. Later there emerged a scheme of *collaborative robotic training* [124], where knowledge from different robots is transferred by sharing their policies and episodic demonstrations with each other. One representative work is [125], which performs policy transfer across multiple robot agents under the *DQN* framework.

Recent work shifted their focus to fast and robust adaptation to unseen tasks. A typical approach is to design and select multiple source domains for robust training, in order to learn a generalized policy from those source tasks, which can be quickly adapted to target domains. Examples include the *EPOPT* algorithm [126], which is a combination of policy transfer via source domain ensemble and learning from limited demonstrations. Another application can be found in [127], in which robust agent policies are trained using a large number of synthetic demonstrations from a simulator to handle dynamic environments. Another solution to fast adaptation is to learn latent representations from observations in the source domain that are generally applicable to the target domain, e.g. training robots using simulated 2D image inputs and applying the robot in real 3D environments. Work along this line includes [128], which learns the latent representation using 3D CAD models, and [129, 130] which are derived based on the Generative-Adversarial Network. Another example is *DARLA* [131], which is a zero-shot transfer approach to learn disentangled representations that are robust against domain shifts.

**Game Playing** is one representative testbed for transfer learning and RL algorithms, whose complexity has evolved over the recent decades, from classical testbeds such as grid-world games to more complex game settings such as online-strategy games or video games with pixel GRB inputs. A representative application in game playing is *AlphaGo*, which is an algorithm for learning the online chessboard games using both transfer learning and RL techniques [73]. *AlphaGo* is first pre-trained offline using expert demonstrations and then learns to optimize its policy using Monte-Carlo Tree Search. Its successor, *AlphaGo Master* [132], even beat the world's first ranked human player.

In addition to online chessboard games, transfer learning approaches have also performed well in video game playing using RL backbones. State-of-the-art video game platforms include *MineCraft*, *Atari*, and *Starcraft*. Especially, [133] designed new RL tasks under the *MineCraft* platform for a better comparison of different RL algorithms. We refer readers to [134] for a survey of AI for real-time strategy (RTS) games on the *Starcraft* platform, with a dataset available from [135]. Moreover, [136] provided a comprehensive survey on DL

applications in video game playing, which also covers transfer learning and RL strategies from certain perspectives. A large portion of transfer learning approaches reviewed in this survey have been applied to the *Atari* [137] and other game above-mentioned platforms. Especially, OpenAI trained an *Dota2* agent that can surpass human experts [138].

**Natural Language Processing (NLP)** has evolved rapidly along with the advancement of DL and RL. Applications of RL on NLP range widely, from *Question Answering (QA)* [139], *Dialogue systems* [140], *Machine Translation* [141], to an integration of NLP and Computer Vision tasks, such as *Visual Question Answering (VQA)* [142], *Image Caption* [143], etc. Many NLP applications have implicitly applied transfer learning approaches, including *learning from demonstrations*, *policy transfer*, or *reward shaping*, to better tailor these RL techniques as NLP solutions, which were previously dominated by the supervise-learning counterparts [144]. Examples in this field include applying expert demonstration to build RL solutions for *Spoken Dialogue Systems* [145], *VQA* [142]; or building shaped rewards for *Sequence Generation* [146], *Spoken Dialogue Systems* [66], *QA* [67, 147], and *Image Caption* [143], or transferring policies for *Structured Prediction* [148] and *VQA* [149], etc.

**Health Informatics** is another domain that has benefited from the advancement of RL. RL has been applied to solve various healthcare tasks, including *dynamic treatment regimes* [150, 151], *automatic medical diagnosis* [152, 153], *health resource scheduling* [154, 155], and *drug discovery and development*, [156, 157], etc. An overview of recent achievements along this line is provided by [158]. Despite the emergence of RL to address healthcare problems, only a limited number of them have utilized transfer learning approaches, although we do observe some applications that leverage prior knowledge to improve the RL procedure. Specifically, [159] utilized *Q*-learning for drug delivery individualization. They integrated the prior knowledge of the dose-response characteristics into their *Q*-learning framework to avoid random exploration. Some work considered reusing representations for speeding up learning [160, 161]. For example, [160] proposed to highlight both the individual variability and the common policy model structure for individual HIV treatment. [161] applied a *DQN* framework for prescribing effective HIV treatments, in which they learned a latent representation to estimate the uncertainty when transferring a pertained policy to the unseen domains. [162] considered the possibility of applying human-involved interactive RL training for health informatics. In general, transfer learning combined with RL is a promising integration for health informatics to improve the learning effectiveness and sample efficiency, especially given the difficulty of accessing large amounts of clinical data.

**Others:** RL has also been utilized in many other real-life applications. Applications in the *Transportation Systems* have adopted RL to address traffic congestion issues with better *traffic signal scheduling* and *transportation resource allocation* [8, 9, 163, 164]. We refer readers to [165] for a review along this line. Deep RL are also effective solutions to problems in *Finance*, including *portfolio management* [166, 167], *asset allocation* [168], and *trading optimization* [169]. Another application is the *Electricity Systems*, especially the *intelligent electricity networks*, which can benefit from RL techniques

Citation	Representations format	Assumptions	MDP difference	Learner	Metrics
[106]	Lateral connections to previously learned network modules	N/A	$\mathcal{S}, \mathcal{A}$	A3C	<i>ap, ps</i>
[107]	Selected neural paths	N/A	$\mathcal{S}, \mathcal{A}$	A3C	<i>ap</i>
[109]	Task(agent)-specific network module	Disentangled state representation	$\mathcal{S}, \mathcal{A}$	Policy Gradient	<i>ap</i>
[111]	Dynamic transitions module learned on state latent representations.	N/A	$\mathcal{S}, \mathcal{A}$	A3C	<i>ap, pe</i>
[94]	SF	Reward function can be linearly decoupled	$\mathcal{R}$	DQN	<i>ap, ar</i>
[113]	Encoder-decoder learned SF	N/A	$\mathcal{R}$	DQN	<i>pe, ps</i>
[54]	Encoder-decoder learned SF	Rewards can be represented by set of basis functions	$\mathcal{R}$	$Q(\lambda)$	<i>ap, pe</i>
[49]	Matrix-factorized UF	Goal conditinoal RL	$\mathcal{R}$	Tabular Q-learning	<i>ap, pe, ps</i>
[117]	Policy-encoded UF	Reward function can be linearly decoupled	$\mathcal{R}$	$\epsilon$ -greedy Q-learning	<i>ap, pe</i>

TABLE 5: A comparison of transfer learning approaches of *representation transfer*.

for improved power-delivery decisions [170, 171] and active resource management [172]. [7] provides a detailed survey of RL techniques for electric power system applications. We believe that given a plethora of theoretical and application breakthroughs, it is promising to embrace a hybrid framework for further accelerating the development of transfer learning in the context of RL.

## 6 FUTURE PERSPECTIVES

In this section, we propose some future directions for the research of transfer learning in the RL domain:

**Evaluating Transferability:** Evaluation metrics and benchmarks have been proposed to review transfer learning approaches from different but complementary perspectives, although no single metric can summarize the efficacy of a transfer learning approach. Designing a set of generalized, novel metrics is beneficial for the development of transfer learning in deep RL. In addition to the current benchmarks, such as OpenAI gym that is designed purely for evaluating RL approaches, a unified benchmark to evaluate the transfer learning performance is worth research and engineering efforts.

**Framework-Agnostic Transfer:** Most contemporary transfer learning approaches are designed for certain RL frameworks. For example, some transfer learning methods are applicable to RL algorithms designed for the discrete-action space (such as DQfD), while others may only be feasible given a continuous action space. One fundamental cause of these framework-dependent transfer learning methods is the diversified development of RL algorithms. We expect that a more unified RL framework would contribute to the standardization of transfer learning approaches in this field.

**Interpretability:** Deep learning and end-to-end systems have made network representation a black box, making it difficult to interpret the model’s reasoning. As a result, there have been efforts in defining and evaluating the *interpretability* of approaches in the supervise-learning domain [173], [174]. The merits of *interpretability* are manifolds, including enabling disentangled representations, building explainable models, facilitating human-computer interactions, etc. In the meantime, interpretable transfer learning approaches for the RL domain, especially with explainable representations or policy decisions, can also be beneficial to many applied

fields, such as *robotics* and *finance*. Moreover, *interpretability* can also help in avoiding catastrophic decision-making for tasks such as auto-driving or healthcare decisions. Although there have been efforts towards explainable transfer learning approaches for RL tasks [175, 176], there is no clear definition for interpretable transfer learning in the context of RL, nor a framework to evaluate the interpretability of transfer learning approaches. We believe that interpretable transfer learning for RL will be a topic that is worth more research efforts in the near future.

## REFERENCES

- [1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A brief survey of deep reinforcement learning,” *arXiv preprint arXiv:1708.05866*, 2017.
- [3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research*, 2016.
- [4] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection,” *The International Journal of Robotics Research*, 2018.
- [5] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, “The arcade learning environment: An evaluation platform for general agents,” *Journal of Artificial Intelligence Research*, 2013.
- [6] M. R. Kosorok and E. E. Moodie, *Adaptive TreatmentStrategies in Practice: Planning Trials and Analyzing Data for Personalized Medicine*. SIAM, 2015.
- [7] M. Glavic, R. Fonteneau, and D. Ernst, “Reinforcement learning for electric power system decision and control: Past considerations and perspectives,” *IFAC-PapersOnLine*, 2017.
- [8] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, “Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto,” *IEEE Transactions on Intelligent Transportation Systems*, 2013.
- [9] H. Wei, G. Zheng, H. Yao, and Z. Li, “Intellilight: A reinforcement learning approach for intelligent traffic

- light control," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.
- [10] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, 2009.
- [11] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, 2009.
- [12] A. Lazaric, "Transfer in reinforcement learning: a framework and a survey." Springer, 2012.
- [13] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, 1957.
- [14] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, England, 1994.
- [15] H. Van Seijen, H. Van Hasselt, S. Whiteson, and M. Wiering, "A theoretical and empirical analysis of expected sarsa," *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2009.
- [16] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, 2000.
- [17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *ICML*, 2016.
- [18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *International Conference on Machine Learning*, 2018.
- [19] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, 1992.
- [20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.
- [21] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," *AAAI*, 2018.
- [22] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, 1992.
- [23] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," *ICML*, 2015.
- [24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [25] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.
- [26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
- [27] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.
- [28] J. Ho and S. Ermon, "Generative adversarial imitation learning," *NeurIPS*, 2016.
- [29] Z. Zhu, K. Lin, B. Dai, and J. Zhou, "Off-policy imitation learning from observations," *NeurIPS*, 2020.
- [30] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, "Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning," *arXiv preprint arXiv:1809.02925*, 2018.
- [31] M. Jing, X. Ma, W. Huang, F. Sun, C. Yang, B. Fang, and H. Liu, "Reinforcement learning from imperfect demonstrations under soft expert guidance," *AAAI*, 2020.
- [32] Z. Zhu, K. Lin, B. Dai, and J. Zhou, "Learning sparse rewarded tasks from sub-optimal demonstrations," *arXiv preprint arXiv:2004.00530*, 2020.
- [33] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.
- [34] R. S. Sutton, A. Koop, and D. Silver, "On the role of tracking in stationary environments," *Proceedings of the 24th international conference on Machine learning*, 2007.
- [35] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," *ICLR*, 2018.
- [36] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
- [37] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," *NeurIPS*, 1993.
- [38] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, 1999.
- [39] R. Parr and S. J. Russell, "Reinforcement learning with hierarchies of machines," *NeurIPS*, 1998.
- [40] T. G. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *Journal of artificial intelligence research*, 2000.
- [41] R. B. Myerson, *Game theory*. Harvard university press, 2013.
- [42] L. Bu, R. Babu, B. De Schutter *et al.*, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2008.
- [43] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," 1993.
- [44] F. L. Da Silva and A. H. R. Costa, "A survey on transfer learning for multiagent reinforcement learning systems," *Journal of Artificial Intelligence Research*, 2019.
- [45] B. Kim, A.-m. Farahmand, J. Pineau, and D. Precup, "Learning from limited demonstrations," *NeurIPS*, 2013.
- [46] W. Czarnecki, R. Pascanu, S. Osindero, S. Jayakumar, G. Swirszcz, and M. Jaderberg, "Distilling policy distillation," *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019.
- [47] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," *ICML*, 1999.
- [48] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio,

- "Generative adversarial nets," *NeurIPS*, pp. 2672–2680, 2014.
- [49] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," *ICML*, 2015.
- [50] C. Finn and S. Levine, "Meta-learning: from few-shot learning to rapid reinforcement learning," *ICML*, 2019.
- [51] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.
- [52] Y. Gao, J. Lin, F. Yu, S. Levine, T. Darrell *et al.*, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.
- [53] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning," *Journal of Machine Learning Research*, 2007.
- [54] A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Žídek, and R. Munos, "Transfer in deep reinforcement learning using successor features and generalised policy improvement," *arXiv preprint arXiv:1901.10964*, 2019.
- [55] R. J. Williams and L. C. Baird, "Tight performance bounds on greedy policies based on imperfect value functions," *Tech. Rep.*, 1993.
- [56] E. Wiewiora, G. W. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," *ICML*, 2003.
- [57] S. M. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," *ICAAMAS*, 2012.
- [58] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowé, "Expressing arbitrary reward functions as potential-based advice," *AAAI*, 2015.
- [59] T. Brys, A. Harutyunyan, M. E. Taylor, and A. Nowé, "Policy transfer using reward shaping," *ICAAMAS*, 2015.
- [60] M. Večerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.
- [61] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer, "Potential-based difference rewards for multiagent reinforcement learning," *ICAAMAS*, 2014.
- [62] M. Grzes and D. Kudenko, "Learning shaping rewards in model-based reinforcement learning," *Proc. AAMAS Workshop on Adaptive Learning Agents*, 2009.
- [63] O. Marom and B. Rosman, "Belief reward shaping in reinforcement learning," *AAAI*, 2018.
- [64] M. Klissarov and D. Precup, "Reward propagation using graph convolutional networks," *NeurIPS*, 2020.
- [65] A. C. Tenorio-Gonzalez, E. F. Morales, and L. Vilaseñor-Pineda, "Dynamic reward shaping: Training a robot by voice," *Advances in Artificial Intelligence – IBERAMIA*, 2010.
- [66] P.-H. Su, D. Vandyke, M. Gasic, N. Mrksic, T.-H. Wen, and S. Young, "Reward shaping with recurrent neural networks for speeding up on-line policy learning in spoken dialogue systems," *arXiv preprint arXiv:1508.03391*, 2015.
- [67] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," *arXiv preprint arXiv:1808.10568*, 2018.
- [68] F. Liu, Z. Ling, T. Mu, and H. Su, "State alignment-based imitation learning," *arXiv preprint arXiv:1911.10947*, 2019.
- [69] K. Kim, Y. Gu, J. Song, S. Zhao, and S. Ermon, "Domain adaptive imitation learning," *ICML*, 2020.
- [70] Y. Ma, Y.-X. Wang, and B. Narayanaswamy, "Imitation-regularized offline learning," *International Conference on Artificial Intelligence and Statistics*, 2019.
- [71] M. Yang and O. Nachum, "Representation matters: Offline pretraining for sequential decision making," *arXiv preprint arXiv:2102.05815*, 2021.
- [72] X. Zhang and H. Ma, "Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations," *arXiv preprint arXiv:1801.10459*, 2018.
- [73] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, 2016.
- [74] S. Schaal, "Learning from demonstration," *NeurIPS*, 1997.
- [75] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," *AAAI*, 2018.
- [76] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [77] J. Chemali and A. Lazaric, "Direct policy iteration with demonstrations," *International Joint Conference on Artificial Intelligence*, 2015.
- [78] B. Piot, M. Geist, and O. Pietquin, "Boosted bellman residual minimization handling expert demonstrations," *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2014.
- [79] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," *International Joint Conference on Artificial Intelligence*, 2015.
- [80] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," *ICML*, 2018.
- [81] D. P. Bertsekas, "Approximate policy iteration: A survey and some new methods," *Journal of Control Theory and Applications*, 2011.
- [82] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *ICLR*, 2016.
- [83] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," *AISTATS*, 2011.
- [84] K. Brantley, W. Sun, and M. Henaff, "Disagreement-regularized imitation learning," *ICLR*, 2019.
- [85] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *Deep Learning and Representation Learning Workshop, NeurIPS*, 2014.
- [86] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *arXiv preprint arXiv:1802.05668*, 2018.
- [87] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation,"

- arXiv preprint arXiv:1511.06295*, 2015.
- [88] H. Yin and S. J. Pan, "Knowledge transfer for deep reinforcement learning with hierarchical experience replay," *AAAI*, 2017.
- [89] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342*, 2015.
- [90] S. Schmitt, J. J. Hudson, A. Zidek, S. Osindero, C. Doersch, W. M. Czarnecki, J. Z. Leibo, H. Kuttler, A. Zisserman, K. Simonyan *et al.*, "Kickstarting deep reinforcement learning," *arXiv preprint arXiv:1803.03835*, 2018.
- [91] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, "Distral: Robust multitask reinforcement learning," *NeurIPS*, 2017.
- [92] J. Schulman, X. Chen, and P. Abbeel, "Equivalence between policy gradients and soft q-learning," *arXiv preprint arXiv:1704.06440*, 2017.
- [93] F. Fernández and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 2006.
- [94] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," *NuerIPS*, 2017.
- [95] R. Bellman, "Dynamic programming," *Science*, 1966.
- [96] L. Torrey, T. Walker, J. Shavlik, and R. Maclin, "Using advice to transfer knowledge acquired in one reinforcement learning task to another," *European Conference on Machine Learning*, 2005.
- [97] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *ICLR*, 2017.
- [98] G. Konidaris and A. Barto, "Autonomous shaping: Knowledge transfer in reinforcement learning," *ICML*, 2006.
- [99] H. B. Ammar and M. E. Taylor, "Reinforcement learning transfer via common subspaces," *Proceedings of the 11th International Conference on Adaptive and Learning Agents*, 2012.
- [100] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, 2017.
- [101] C. Wang and S. Mahadevan, "Manifold alignment without correspondence," *International Joint Conference on Artificial Intelligence*, 2009.
- [102] B. Boci, L. Csató, and J. Peters, "Alignment-based transfer learning for robot models," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013.
- [103] H. B. Ammar, E. Eaton, P. Ruvolo, and M. E. Taylor, "Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment," *AAAI*, 2015.
- [104] H. B. Ammar, K. Tuyls, M. E. Taylor, K. Driessens, and G. Weiss, "Reinforcement learning transfer via sparse coding," *ICAAMS*, 2012.
- [105] A. Lazaric, M. Restelli, and A. Bonarini, "Transfer of samples in batch reinforcement learning," *ICML*, 2008.
- [106] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.
- [107] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.
- [108] I. Harvey, "The microbial genetic algorithm," *European Conference on Artificial Life*, 2009.
- [109] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [110] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," *ICML*, 2017.
- [111] A. Zhang, H. Satija, and J. Pineau, "Decoupling dynamics and reward for transfer learning," *arXiv preprint arXiv:1804.10689*, 2018.
- [112] P. Dayan, "Improving generalization for temporal difference learning: The successor representation," *Neural Computation*, 1993.
- [113] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman, "Deep successor reinforcement learning," *arXiv preprint arXiv:1606.02396*, 2016.
- [114] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.
- [115] N. Mehta, S. Natarajan, P. Tadepalli, and A. Fern, "Transfer in variable-reward hierarchical reinforcement learning," *Machine Learning*, 2008.
- [116] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE transactions on information theory*, 2010.
- [117] D. Borsa, A. Barreto, J. Quan, D. Mankowitz, R. Munos, H. van Hasselt, D. Silver, and T. Schaul, "Universal successor features approximators," *arXiv preprint arXiv:1812.07626*, 2018.
- [118] L. Lehnert, S. Tellex, and M. L. Littman, "Advantages and limitations of using successor features for transfer in reinforcement learning," *arXiv preprint arXiv:1708.00102*, 2017.
- [119] J. C. Petangoda, S. Pascual-Diaz, V. Adam, P. Vranckx, and J. Grau-Moya, "Disentangled skill embeddings for reinforcement learning," *arXiv preprint arXiv:1906.09223*, 2019.
- [120] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *ICML*, 2017.
- [121] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," *ICML*, 2004.
- [122] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, 2013.
- [123] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, 2009.
- [124] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation,"

- IEEE Transactions on automation science and engineering*, 2015.
- [125] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," *IEEE international conference on robotics and automation (ICRA)*, 2017.
- [126] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "Epopt: Learning robust neural network policies using model ensembles," *arXiv preprint arXiv:1610.01283*, 2016.
- [127] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv preprint arXiv:1702.02453*, 2017.
- [128] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.
- [129] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [130] H. Bharadhwaj, Z. Wang, Y. Bengio, and L. Paull, "A data-efficient framework for training and sim-to-real transfer of navigation policies," *International Conference on Robotics and Automation (ICRA)*, 2019.
- [131] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner, "Darla: Improving zero-shot transfer in reinforcement learning," *ICML*, 2017.
- [132] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, 2017.
- [133] J. Oh, V. Chockalingam, S. Singh, and H. Lee, "Control of memory, active perception, and action in minecraft," *arXiv preprint arXiv:1605.09128*, 2016.
- [134] S. Ontanón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A survey of real-time strategy game ai research and competition in starcraft," *IEEE Transactions on Computational Intelligence and AI in games*, 2013.
- [135] Z. Lin, J. Gehring, V. Khalidov, and G. Synnaeve, "Stardata: A starcraft ai research dataset," *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2017.
- [136] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Transactions on Games*, 2019.
- [137] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
- [138] OpenAI. (2019) Dotal2 blog. [Online]. Available: <https://openai.com/blog/openai-five/>
- [139] H. Chen, X. Liu, D. Yin, and J. Tang, "A survey on dialogue systems: Recent advances and new frontiers," *Acm Sigkdd Explorations Newsletter*, 2017.
- [140] S. P. Singh, M. J. Kearns, D. J. Litman, and M. A. Walker, "Reinforcement learning for spoken dialogue systems," *NeurIPS*, 2000.
- [141] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.
- [142] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko, "Learning to reason: End-to-end module networks for visual question answering," *IEEE International Conference on Computer Vision*, 2017.
- [143] Z. Ren, X. Wang, N. Zhang, X. Lv, and L.-J. Li, "Deep reinforcement learning-based image captioning with embedding reward," *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [144] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational intelligenCe magazine*, 2018.
- [145] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Learning to compose neural networks for question answering," *arXiv preprint arXiv:1601.01705*, 2016.
- [146] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, "An actor-critic algorithm for sequence prediction," *arXiv preprint arXiv:1607.07086*, 2016.
- [147] F. Godin, A. Kumar, and A. Mittal, "Learning when not to answer: a ternary reward structure for reinforcement learning based question answering," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.
- [148] K.-W. Chang, A. Krishnamurthy, A. Agarwal, J. Langford, and H. Daumé III, "Learning to search better than your teacher," 2015.
- [149] J. Lu, A. Kannan, J. Yang, D. Parikh, and D. Batra, "Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model," *NeurIPS*, 2017.
- [150] E. B. Laber, D. J. Lizotte, M. Qian, W. E. Pelham, and S. A. Murphy, "Dynamic treatment regimes: Technical challenges and applications," *Electronic journal of statistics*, 2014.
- [151] M. Tenenbaum, A. Fern, L. Getoor, M. Littman, V. Mansinghka, S. Natarajan, D. Page, J. Shrager, Y. Singer, and P. Tadepalli, "Personalizing cancer therapy via machine learning," *Workshops of NeurIPS*, 2010.
- [152] A. Alansary, O. Oktay, Y. Li, L. Le Folgoc, B. Hou, G. Vaillant, K. Kamnitsas, A. Vlontzos, B. Glocker, B. Kainz *et al.*, "Evaluating reinforcement learning agents for anatomical landmark detection," 2019.
- [153] K. Ma, J. Wang, V. Singh, B. Tameroy, Y.-J. Chang, A. Wimmer, and T. Chen, "Multimodal image registration with deep context reinforcement learning," *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2017.
- [154] Z. Huang, W. M. van der Aalst, X. Lu, and H. Duan, "Reinforcement learning based resource allocation in business process management," *Data & Knowledge Engineering*, 2011.
- [155] T. S. M. T. Gomes, "Reinforcement learning for primary care appointment scheduling," 2017.
- [156] A. Serrano, B. Imbernón, H. Pérez-Sánchez, J. M. Cecilia, A. Bueno-Crespo, and J. L. Abellán, "Accelerating

- drugs discovery with deep reinforcement learning: An early approach," *International Conference on Parallel Processing Companion*, 2018.
- [157] M. Popova, O. Isayev, and A. Tropsha, "Deep reinforcement learning for de novo drug design," *Science advances*, 2018.
- [158] C. Yu, J. Liu, and S. Nemati, "Reinforcement learning in healthcare: A survey," *arXiv preprint arXiv:1908.08796*, 2019.
- [159] A. E. Gaweda, M. K. Muezzinoglu, G. R. Aronoff, A. A. Jacobs, J. M. Zurada, and M. E. Brier, "Incorporating prior knowledge into q-learning for drug delivery individualization," *Fourth International Conference on Machine Learning and Applications*, 2005.
- [160] V. N. Marivate, J. Chemali, E. Brunskill, and M. Littman, "Quantifying uncertainty in batch personalized sequential decision making," *Workshops at the Twenty-Eighth AAAI*, 2014.
- [161] T. W. Killian, S. Daulton, G. Konidaris, and F. Doshi-Velez, "Robust and efficient transfer learning with hidden parameter markov decision processes," *NeurIPS*, 2017.
- [162] A. Holzinger, "Interactive machine learning for health informatics: when do we need the human-in-the-loop?" *Brain Informatics*, 2016.
- [163] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, 2016.
- [164] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.
- [165] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, "A survey on reinforcement learning models and algorithms for traffic signal control," *ACM Computing Surveys (CSUR)*, 2017.
- [166] J. Moody, L. Wu, Y. Liao, and M. Saffell, "Performance functions and reinforcement learning for trading systems and portfolios," *Journal of Forecasting*, 1998.
- [167] Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning," *IEEE Intelligent Systems Conference (IntelliSys)*, 2017.
- [168] R. Neuneier, "Enhancing q-learning for optimal asset allocation," *NeurIPS*, 1998.
- [169] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems*, 2016.
- [170] G. Dalal, E. Gilboa, and S. Mannor, "Hierarchical decision making in electricity grid management," *International Conference on Machine Learning*, 2016.
- [171] F. Ruelens, B. J. Claessens, S. Vandael, B. De Schutter, R. Babuška, and R. Belmans, "Residential demand response of thermostatically controlled loads using batch reinforcement learning," *IEEE Transactions on Smart Grid*, 2016.
- [172] Z. Wen, D. O'Neill, and H. Maei, "Optimal demand response using device-based reinforcement learning," *IEEE Transactions on Smart Grid*, 2015.
- [173] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," *IEEE 5th International Conference on data science and advanced analytics (DSAA)*, 2018.
- [174] Q.-s. Zhang and S.-C. Zhu, "Visual interpretability for deep learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, 2018.
- [175] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," *NeurIPS*, 2017.
- [176] R. Ramakrishnan and J. Shah, "Towards interpretable explanations for transfer learning in sequential tasks," *AAAI Spring Symposium Series*, 2016.



**Zhuangdi Zhu** received her bachelor's degree in Computer Science from the College of Elite Education at Nanjing University of Science and Technology, in 2015. She is currently a Ph.D. candidate at the Computer Science department of Michigan State University, MI, USA. Her current research interests reside in the general area of Machine Learning and its practical applications.



**Kaixiang Lin** is an applied scientist at Amazon web services. He obtained his Ph.D. from Michigan State University. He has broad research interests across multiple fields, including reinforcement learning, human-robot interactions, and natural language processing. His research has been published on multiple top-tiered machine learning and data mining conferences such as ICLR, KDD, NeurIPS, etc. He serves as a reviewer for top machine learning conferences regularly.



**Anil K. Jain** is a University distinguished professor in the Department of Computer Science and Engineering at Michigan State University. His research interests include pattern recognition and biometric authentication. He served as the editor-in-chief of the IEEE Transactions on Pattern Analysis and Machine Intelligence and was a member of the United States Defense Science Board. He has received Fulbright, Guggenheim, Alexander von Humboldt, and IAPR King Sun Fu awards. He is a member of the National Academy of Engineering and a foreign fellow of the Indian National Academy of Engineering and the Chinese Academy of Sciences.



**Jiayu Zhou** is currently an associate professor in the Department of Computer Science and Engineering at Michigan State University. He received his Ph.D. degree in computer science from Arizona State University in 2014. He has broad research interests in the fields of large-scale machine learning and data mining as well as biomedical informatics. He has served as a technical program committee member for premier conferences such as NIPS, ICML, and SIGKDD. His papers have received the Best Student Paper Award at the 2014 IEEE International Conference on Data Mining (ICDM), the Best Student Paper Award at the 2016 International Symposium on Biomedical Imaging (ISBI) and the Best Paper Award at IEEE Big Data 2016.