Transfer Learning in Deep Reinforcement Learning: A Survey

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

(Survey Paper)

***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.**

I. 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 deep learning research, the combination of RL 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* (DRL) [2], which learns powerful function approximators using deep neural networks to address complicated domains. DRL has achieved notable success in applications such as robotics control [3], [4] and game playing [5]. It also thrives 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 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 or even incur safety concerns for domains such as automatic-driving and health informatics. The abovementioned challenges have motivated various efforts to improve the current RL procedure. As a result, *transfer learning* (TL), or equivalently referred as *knowledge transfer*, which is a technique to utilize external expertise to benefit the learning process of the target domain, becomes a crucial topic in RL.

While TL techniques have been extensively studied in *supervised learning* [10], it is still an emerging topic for RL. Transfer learning can be more complicated for RL, 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 that need to transfer in different ways. Noticing that previous efforts on summarizing TL in the RL domain did not cover research of the last decade [11], [12], during which time considerate TL breakthroughs have been achieved empowered with deep learning techniques. Hence, in this survey, we make a comprehensive investigation of the latest TL approaches in RL.

The contributions of our survey are multifold: 1) we investigated up-to-date research involving new DRL backbones and TL algorithms over the recent decade. To the best of our knowledge, this survey is the first attempt to survey TL approaches in the context of ***deep*** reinforcement learning. We reviewed TL methods that can tackle more evolved RL tasks, and also studied new TL schemes that are not deeply discussed by prior literatures, such as representation disentanglement (Sec V-E) and policy distillation (Sec V-C). 2) We provided systematic categorizations that cover a broader and deeper view of TL developments in DRL. Our main analysis is anchored on a fundamental question, i.e., *what is the transferred knowledge in*

Manuscript received 1 February 2022; revised 25 May 2023; accepted 23 June 2023. Date of publication 4 July 2023; date of current version 3 October 2023. This work was supported in part by the National Science Foundation under Grants IIS-2212174 and IIS-1749940, in part by the National Institute of Aging under Grant IRF1AG072449, and in part by the Office of Naval Research under Grant N00014-20-1-2382. Recommended for acceptance by M. White. *(Corresponding authors: Zhuangdi Zhu; Jiayu Zhou.)*

The authors are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: zhuzhuan@msu.edu; kaixianglin.cs@gmail.com; jain@msu.edu; jiayuz@msu.edu).

Digital Object Identifier 10.1109/TPAMI.2023.3292075

RL, following which we conducted more refined analysis. Most TL strategies, including those discussed in prior surveys are well suited in our categorization framework. 3) Reflecting on the developments of TL methods in DRL, we brought new thoughts on its future directions, including how to do *reasoning* over miscellaneous knowledge forms and how to *leverage* knowledge in more efficient and principled manner. We also pointed out the prominent applications of TL for DRL and its opportunities to thrive in the future era of AGI.

The rest of this survey is organized as follows: In Section II we introduce RL preliminaries, including the recent key development based on deep neural networks. Next, we discuss the definition of TL in the context of RL and its relevant topics (Section II-D). In Section III, we provide a framework to categorize TL approaches from multiple perspectives, analyze their fundamental differences, and summarize their evaluation metrics (Section III-C). In Section V, we elaborate on different TL approaches in the context of DRL, organized by the format of transferred knowledge, such as *reward shaping* (Section V-A), *learning from demonstrations* (Section V-B), or *learning from teacher policies* (Section V-C). We also investigate TL approaches by the way that knowledge transfer occurs, such as inter-task mapping (Section V-D), or learning transferrable representations (Section V-E), etc. We discuss contemporary applications of TL in the context of DRL in Section VI and provide some future perspectives and open questions in Section VII.

II. Deep Reinforcement Learning and Transfer Learning

A. Reinforcement Learning Basics

Markov Decision Process: A typical RL problem can be considered as training an agent to interact with an environment that follows a *Markov Decision Process* (MPD) [13]. 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. A subsequence of interactions is referred to as an *episode*. The above-mentioned components in an MDP can be represented using a tuple, i.e., $\mathcal{M} = (\mu_0, \mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R})$, in which:

- μ_0 is the set of **initial states**.
- $\mathcal{S}$ is the **state** space.
- $\mathcal{A}$ is the **action** space.
- $\mathcal{T}$: $S \times A \times S \to \mathbb{R}$ 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.
- $\mathcal{R} : S \times A \times S \to \mathbb{R}$ 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'.
- γ is a discounted factor, with $\gamma \in (0,1]$.

A RL agent behaves in $\mathcal{M}$ by following its policy π, which is a mapping from states to actions: $\pi : \mathcal{S} \to \mathcal{A}$. For a stochastic policy π, $\pi(a|s)$ denotes the probability of taking action a from state s. Given an MDP $\mathcal{M}$ and a policy π, one can derive a ***value function*** $V^\pi_\mathcal{M}(s)$, which is defined over the state space: $V^\pi_\mathcal{M}(s) = \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots; \pi, s]$, 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 following policy π. Similar to the value function, a policy also carries a ***Q*-function**, which estimates the quality of taking action a from state s: $Q^\pi_\mathcal{M}(s,a) = \mathbb{E}_{s' \sim \mathcal{T}(\cdot|s,a)}[\mathcal{R}(s,a,s') + \gamma V^\pi_\mathcal{M}(s')]$.

Reinforcement Learning Goals: Standard RL aims to learn an optimal policy $\pi^*_\mathcal{M}$ with the optimal value and Q-function, $s.t. \forall s \in \mathcal{S}, \pi^*_\mathcal{M}(s) = \arg\max_{a \in A} Q^*_\mathcal{M}(s,a)$, where $Q^*_\mathcal{M}(s,a) = \sup_\pi Q^\pi_\mathcal{M}(s,a)$. The learning objective can be reduced as maximizing the expected return:

$$J(\pi) := \mathbb{E}_{(s,a) \sim \mu^\pi(s,a)} \left[\sum_t \gamma^t r_t \right],$$

where $\mu^\pi(s,a)$ is the *stationary state-action distribution* induced by π [14].

Built upon recent progress of DRL, some literature has extended the RL objective to achieving miscellaneous goals under different conditions, referred to as ***Goal-Conditional RL*** (GCRL). In GCRL, the agent policy $\pi(\cdot|s,g)$ is dependent not only on state observations s but also the goal g being optimized. Each individual goal $g \sim \mathcal{G}$ can be differentiated by its reward function $r(s_t, a_t, g)$, hence the objective for GCRL becomes maximizing the expected return over the distribution of goals: $J(\pi) := \mathbb{E}_{(s_t,a_t) \sim \mu^\pi, g \sim \mathcal{G}}[\sum_t \gamma^t r(s,a,g)]$ [15]. A prototype example of GCRL can be maze locomotion tasks, where the learning goals are manifested as desired locations in the maze [16].

Episodic Versus Non-episodic Reinforcement Learning: In episodic RL, the agent performs in finite episodes of length H, and will be *reset* to an initial state $\in \mu_0$ upon the episode ends [1]. Whereas in non-episodic RL, the learning agent continuously interacts with the MDP without any state reset [17]. To encompass the episodic concept in infinite MDPs, episodic RL tasks usually assume the existence of a set of *absorbing states* $\mathcal{S}_0$, which indicates the termination of episodic tasks [18], [19], and any action taken upon an absorbing state will only transit to itself with zero rewards.

B. Reinforcement Learning Algorithms

There are two major methods to conduct RL: ***Model-Based*** and ***Model-Free***. In *model-based* RL, a learned or provided model of the MDP is used for policy learning. In *model-free* RL, optimal policy is learned without modeling the transition dynamics or reward functions. In this section, we start introducing RL techniques from a ***model-free*** perspective, due to its relatively simplicity, which also provides foundations for many model-based methods.

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 iterating through these two steps, known as *policy iteration*.

For *model-free* policy iterations, the target policy is optimized without requiring knowledge of the MDP transition dynamics. Traditional model-free RL includes ***Monte-Carlo*** methods, which estimates the value of each state using *samples of 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* [20], *Expected SARSA* [21], *Actor-Critic* [22], and its deep neural network extension called *A3C* [23]. The *off-policy* TD-learning approaches include SAC [24] for continuous state-action spaces, and Q-learning [25] for discrete state-action spaces, along with its variants built on deep-neural networks, such as DQN [26], Double-DQN [26], Rainbow [27], etc. TD-learning approaches 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 parameterizable policy. Traditional policy-gradient approaches include *REINFORCE* [28]. 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)* [29], *Proximal Policy optimization (PPO)* [30], *Deterministic policy gradient (DPG)* [31] and its extensions such as *DDPG* [32] and *Twin Delayed DDPG* [33].

Unlike model-free methods that learn purely from trial-and-error, ***Model-Based RL*** (MBRL) explicitly learns the transition dynamics or cost functions of the environment. The dynamics model can sometimes be treated as a ***black-box*** for better *sampling-based planning*. Representative examples include the *Monte-Carlo* method dubbed *random shooting* [34] and its cross-entropy method (CEM) variants [35], [36]. The modeled dynamics can also facilitate learning with data generation [37] and value estimation [38]. For MBRL with ***white-box*** modeling, the transition models become differentiable and can facilitate planning with direct gradient propogation. Methods along this line include *differential planning* for policy gradient [39] and action sequences search [40], and value gradient methods [41], [42]. One advantage of MBRL is its higher sample efficiency than model-free RL, although it can be challenging for complex domains, where it is usually more difficult to learn the dynamics than learning a policy.

C. Transfer Learning in the Context of Reinforcement Learning

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

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

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

where $\pi = \phi(\mathcal{I}_s \sim \boldsymbol{\mathcal{M}}_s, \mathcal{I}_t \sim \mathcal{M}_t) : \mathcal{S}^t \to \mathcal{A}^t$ is a policy learned for the target domain $\mathcal{M}_t$ 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 $\mathcal{I}$, which is usually approximated with deep neural networks in DRL. For the simplistic case, knowledge can transfer between two agents within the same domain, resulting in $|\boldsymbol{\mathcal{M}}_s| = 1$, and $\mathcal{M}_s = \mathcal{M}_t$. One can consider regular RL without TL as a special case of the above definition, by treating $\mathcal{I}_s = \emptyset$, so that a policy π is learned purely on the feedback provided by the target domain, i.e., $\pi = \phi(\mathcal{I}_t)$.

D. Related Topics

In addition to TL, 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 TL by analyzing the differences and connections between transfer learning and these relevant techniques, which we hope can further clarify the scope of this survey.

Continual Learning is the ability of sequentially learning multiple tasks that are temporally or spatially related, without forgetting the previously acquired knowledge. Continual Learning is a specialized yet more challenging scenario of TL, in that the learned knowledge needs to be transferred along a sequence of dynamically-changing tasks that cannot be foreseen, rather than learning a fixed group of tasks. Hence, different from most TL methods discussed in this survey, the ability of *automatic task detection* and *avoiding catastrophic forgetting* is usually indispensable in continual learning [43].

Hierarchical RL has been proposed to resolve complex real-world tasks. Different from traditional RL, for hierarchical RL, the action space is grouped into different granularities to form higher-level macro actions. Accordingly, the learning task is also decomposed into hierarchically dependent sub-goals. Well-known hierarchical RL frameworks include *Feudal learning* [44], *Options framework* [45], *Hierarchical Abstract Machines* [46], and *MAXQ* [47]. Given the higher-level abstraction on tasks, actions, and state spaces, hierarchical RL can facilitate knowledge transfer across similar domains.

Multi-Task RL learns an agent with generalized skills across various tasks, hence it can solve MDPs randomly sampled from a fixed yet unknown distribution [48]. A larger concept of multi-task learning also incorporates multi-task *supervised* learning and *unsupervised* learning [49]. Multi-task learning is naturally related to TL, in that the learned skills, typically manifested as representations, need to be effectively shared among domains. Many TL techniques later discussed in this survey can be readily applied to solve multi-task RL scenarios, such as policy distillation [50], and representation sharing [51]. One notable challenges in multi-task learning is *negative transfer*, which is induced by the irrelevance or conflicting property for learned tasks. Hence, some recent work in multi-task RL focused

on a trade-off between sharing and individualizing function modules [52], [53], [54].

Generalization in RL refers to the ability of learning agents to adapt to *unseen* domains. Generalization is a crucial property for RL to achieve, especially when classical RL assumes identical training and inference MDPs, whereas the real world is constantly changing. Generalization in RL is considered more challenging than in supervised learning due to the non-stationarity of MDPs, where the latter has provided inspirations for the former [55]. ***Meta-learning*** is an effective direction towards generalization, which also draws close connections to TL. Some TL techniques discussed in this survey are actually designed for meta-RL. However, meta-learning is particularly focused on the learning methods that lead to *fast adaptation* to unseen domains, whereas TL is a broader concept and covers scenarios where the target environment can be (partially) observable. To tackle unseen tasks in RL, some meta-RL methods focused on training MDPs generation [56] and variations estimation [57]. We refer readers to [58] for a more focused survey on meta RL.

III. Analyzing Transfer Learning

In this section, we discuss TL approaches in RL from different angles. We also use a prototype to illustrate the potential variants residing in knowledge transfer among domains, then summarize important metrics for TL evaluation.

A. Categorization of Transfer Learning Approaches

TL approaches can be organized by answering the following key questions:

1) *What knowledge is transferred*: Knowledge from the source domain can take different forms, such as expert experiences [59], the action probability distribution of an expert policy [60], or even a potential function that estimates the quality of demonstrations in the target MDP [61]. The divergence in *representations* and *granularities* of knowledge fundamentally influences how TL is performed. The *quality* of the transferred knowledge, e.g., whether it comes from an oracle [62] or a suboptimal teacher [63] also affects the way TL methods are designed.
2) *What RL frameworks fit the TL approach:* We can rephrase this question into other forms, e.g., *is the TL approach policy-agnostic, or only applicable to certain RL backbones, such as the Temporal Difference (TD) methods?* Answers to this question are closely related to the representaion of knowledge. For example, transferring knowledge from expert demonstrations are usually policy-agnostic (see Section V-B), while policy distillation, to be discussed in Section V-C, may not be suitable for DQN backbone which does not explicitly learn a policy function.
3) *What is the difference between the source and the target domain*: Some TL approaches fit 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 goal-conditioned RL [64], the two domains may differ only by the reward distribution: $\mathcal{R}_s \neq \mathcal{R}_t$.
4) *What information is available in the target domain:* While knowledge from source domains is usually accessible, it can be prohibitive to sample from the target domain, or the reward signal can be sparse or delayed. Examples include adapting an auto-driving agent pre-trained in simulated platforms to real environments [65], The accessibility of information in the target domain can affect the way that TL approaches are designed.
5) *How sample-efficient the TL approach is:* TL enables the RL with better initial performance, hence usually requires fewer interactions compared with learning from scratch. Based on the sampling cost, we can categorize TL 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; (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 TL to be more sample efficient compared to normal RL.

B. Case Analysis of Transfer Learning in the Context of Reinforcement Learning

We now use *HalfCheetah*[1] as a working example to illustrate how TL can occur between the source and the target domain. *HalfCheetah* is a standard DRL benchmark for solving physical locomotion tasks, in which the objective is to train a two-leg agent to run fast without losing control of itself.

1) Potential Domain Differences: During TL, the differences between the source and target domain may reside in any component of an MDP:

- $\mathcal{S}$ *(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.
- μ_0 *(Initial states):* the source and target domains may have different initial states, specifying where and with what posture the agent can start moving.
- τ *(Trajectories):* the source and target domains may allow a different number of steps for the agent to move before a task is done.

2) Transferrable Knowledge: Without losing generality, we list below some transferrable knowledge assuming that the source and target domains are variants of *HalfCheetah:*

[1]https://gym.openai.com/envs/HalfCheetah-v2/

- *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 RL agent may access a model of the physical dynamics for the source domain that is also partly applicable to the target domain. It can perform dynamic programming based on the physical rules, running fast without 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.

C. Evaluation Metrics

In this section, we present some representative metrics for evaluating TL approaches, which have also been partly summarized in prior work [11], [66]:

- *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 TL and *asymptotic performance* of the agent without TL.
- *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 (nka):* the necessary *amount* of the knowledge required for TL in order to achieve certain performance thresholds. Examples along this line include the number of designed source tasks [67], the number of expert policies, or the number of demonstrated interactions [68] required to enable knowledge transfer.
- *Necessary knowledge quality (nkq):* the guaranteed *quality* of the knowledge required to enable effective TL. This metric helps in answering questions such as (i) Does the TL approach rely on near-oracle knowledge, such as expert demonstrations/policies [69], or (ii) is the TL technique feasible even given suboptimal knowledge [63]?

TL approaches differ in various perspectives, including the forms of transferred knowledge, the RL frameworks utilized to enable such transfer, and the gaps between the source and the target domain. It maybe biased to evaluate TL from just one viewpoint. We believe that explicating these TL related metrics helps in designing more generalizable and efficient TL approaches.

In general, most of the abovementioned metrics can be considered as evaluating two abilities of a TL 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.

IV. Related Work

There are prior efforts in summarizing TL research in RL. One of the earliest literatures is [11]. Their main categorization is from the perspective of *problem setting*, in which the TL scenarios may vary in the number of domains involved, and the difference of state-action space among domains. Similar categorization is adopted by [12], with more refined analysis dimensions including the objective of TL. As pioneer surveys for TL in RL, neither [11] nor [12] covered recent research over the last decade. For instance, [11] emphasized on different *task-mapping* methods, which are more suitable for domains with tabular or mild state-action space dimensions.

There are other surveys focused on specific subtopics that interplay between RL and TL. For instance, [70] consolidated sim-to-real TL methods. They explored work that is more tailored for *robotics* domains, including domain generalization and zero-shot transfer, which is a favored application field of DRL as we discussed in Sec VI. [71] conducted extensive database search and summarized benchmarks for evaluating TL algorithms in RL. [72] surveyed recent progress in multi-task RL. They partially shared research focus with us by studying certain TL oriented solutions towards multi-task RL, such as learning shared representations, pathNets, etc. We surveyed TL for RL with a broader spectrum in methodologies, applications, evaluations, which naturally draws connections to the above literatures.

V. Transfer Learning Approaches Deep Dive

In this section, we elaborate on various TL approaches and organize them into different sub-topics, mostly by answering the question of *"what knowledge is transferred"*. For each type of TL approach, we analyze them by following the other criteria mentioned in Section III and and summarize the key evaluation metrics that are applicable to the discussed work. Fig. 1 presents an overview of different TL approaches discussed in this survey.

A. Reward Shaping

We start by introducing the *Reward Shaping* approach, as it is applicable to most RL backbones and also largely overlaps with the other TL approaches discussed later. 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

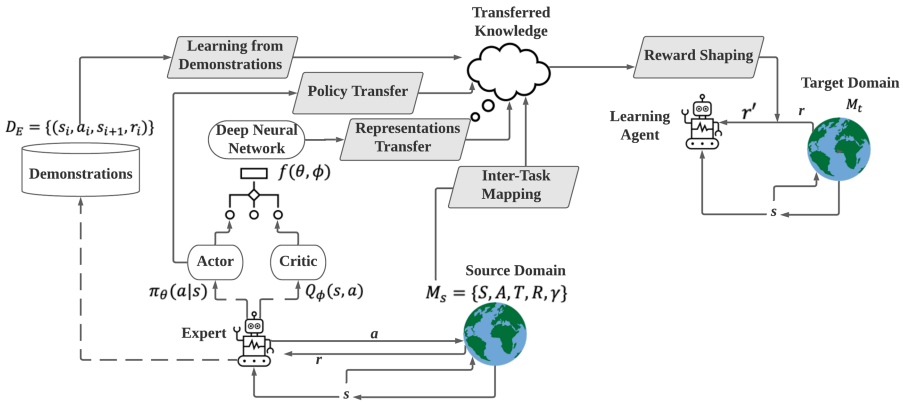

Fig. 1. An overview of different TL approaches, organized by the format of transferred knowledge.

environment reward signals, RS learns a reward-shaping function $\mathcal{F} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ 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 to 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:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R})) \to \mathcal{M}' = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R}'). \quad (1)$$

Along the line of RS, *Potential based Reward Shaping (PBRS)* is one of the most classical approaches. [61] proposed PBRS to form a shaping function F as the difference between two *potential functions* ($\Phi(\cdot)$):

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s), \quad (2)$$

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: $Q^*_{\mathcal{M}'}(s, a) = Q^*_{\mathcal{M}}(s, a) - \Phi(s)$, which draws a connection between potential based reward-shaping and advantage-based learning approaches [73].

The idea of *PBRS* was extended to [74], 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:

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

PBA requires on-policy learning and can be *sample-costly*, as in Equation (3), a' is the action to take upon state s is transitioning to s' by following the learning policy.

Traditional RS approaches assumed a *static* potential function, until [75] 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 tilmestep. [76] later introduced a way to incorporate any prior knowledge into a dynamic potential function structure, which is called ***Dynamic Value Function Advice (DPBA)***. The rationale behind 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 Φ is not static but learned as an extra state-action *Value* function overtime, then the Bellman equation for Φ is : $\Phi^\pi(s, a) = r^\Phi(s, a) + \gamma\Phi(s', a')$. The shaping rewards $F(s, a)$ is therefore the negation of $r^\Phi(s, a)$:

$$F(s, a) = \gamma\Phi(s', a') - \Phi(s, a) = -r^\Phi(s, a). \quad (4)$$

This leads to the approach of using the negation of R^+ as the immediate reward to train an extra state-action *Value* function Φ and the policy simultaneously. Accordingly, the dynamic potential function F becomes:

$$F_t(s, a) = \gamma\Phi_{t+1}(s', a') - \Phi_t(s, a). \quad (5)$$

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

Research along this line mainly focus on designing different shaping functions $F(s, a)$, while not much work has tackled the question of *what knowledge can be used to derive this potential function*. One work by [77] proposed to use RS to transfer an expert policy from the source domain $\mathcal{M}_s$ to the target domain $\mathcal{M}_t$. This approach assumed the existence of two mapping functions M_S and M_A that can transform the state and action from the source to the target domain. Another work used demonstrated state-action samples from an expert policy to shape rewards [78]. Learning the augmented reward involves learning a discriminator 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 combines two TL approaches: RS and *Learning from Demonstrations*, the latter of which will be elaborated in Section V-B.

TABLE I
A COMPARISON OF REWARD SHAPING APPROACHES

Methods	MDP difference	Format of shaping reward	Knowledge source	Evaluation metrics
PBRS	$\mathcal{M}_s = \mathcal{M}_t$	$F = \gamma\Phi(s') - \Phi(s)$	✗	*ap, ar*
PBA	$\mathcal{M}_s = \mathcal{M}_t$	$F = \gamma\Phi(s', a') - \Phi(s, a)$	✗	*ap, ar*
DPB	$\mathcal{M}_s = \mathcal{M}_t$	$F = \gamma\Phi(s', t') - \Phi(s, t)$	✗	*ap, ar*
DPBA	$\mathcal{M}_s = \mathcal{M}_t$	$F_t = \gamma\Phi_{t+1}(s', a') - \Phi_t(s, a)$, Φ learned as an extra Q function	✗	*ap, ar*
[77]	$\mathcal{S}_s \neq \mathcal{S}_t, \mathcal{A}_s \neq \mathcal{A}_t$	$F_t = \gamma\Phi_{t+1}(s', a') - \Phi_t(s, a)$	π_s	*ap, ar*
[78]	$\mathcal{M}_s = \mathcal{M}_t$	$F_t = \gamma\Phi_{t+1}(s', a') - \Phi_t(s, a)$	D_E	*ap, ar*

✗ Denotes that the information is not revealed in the paper.

The above-mentioned RS approaches are summarized in Table I. They follow the potential based RS principle that has been developed systematically: from the classical *PBRS* 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 TL paradigm, RS has been widely applied to fields including robot training [79], spoken dialogue systems [80], and question answering [81]. It provides a feasible framework for transferring knowledge as the augmented reward and is generally applicable to various RL algorithms. RS has also been applied to *multi-agent* RL [82] and *model-based* RL [83]. Principled integration of RS with other TL approaches, such as *Learning from demonstrations* (Section V-B) and *Policy Transfer* (Section V-C) will be an intriguing question for ongoing research.

Note that 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 RS work also tackles the scenario where the shaped reward comes *intrinsically*. For instance, *Belief Reward Shaping* was proposed by [84], 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.

B. Learning From Demonstrations

Learning from Demonstrations (LfD) is a technique to assist RL by utilizing external demonstrations for more efficient exploration. The demonstrations may come from different sources with varying qualities. Research along this line usually address a scenario where 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 [85], [86].

Depending on ***when*** the demonstrations are used for knowledge transfer, approaches can be organized into *offline* and *online* methods. For *offline* approaches, demonstrations are either used for pre-training RL components, or for *offline* RL [87], [88]. When leveraging demonstrations for pre-training, RL components such as the value function $V(s)$ [89], the policy π [90], or the model of transition dynamics [91], can be initialized by learning from demonstrations. For the *online* approach, demonstrations are directly used to guide agent actions for efficient explorations [92]. Most work discussed in this section follows the online transfer paradigm or combines offline pre-training with online RL [93].

Work along this line can also be categorized depending on **what** RL frameworks are compatible: some adopts the policy-iteration framework [59], [94], [95], some follow a Q-learning framework [92], [96], while recent work usually follows the policy-gradient framework [63], [78], [93], [97]. Demonstrations have been leveraged in the *policy iterations* framework by [98]. Later, [94] introduced the *Direct Policy Iteration with Demonstrations (DPID)* algorithm. This approach samples complete demonstrated rollouts D_E from an expert policy π_E, in combination with the self-generated rollouts D_π 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 π is further regularized by a loss function $\mathcal{L}(s, \pi_E)$ to minimize its discrepancy from the expert policy decision.

Another example is the *Approximate Policy Iteration with Demonstration (APID)* algorithm, which was proposed by [59] and extended by [95]. Different from *DPID* where both D_E and D_π are used for value estimation, the *APID* algorithm solely applies D_π to approximate on the Q function. 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:

$$Q(s_i, \pi_E(s_i)) - \max_{a \in \mathcal{A} \setminus \pi_E(s_i)} Q(s_i, a) \geq 1 - \xi_i. \tag{6}$$

The term ξ_i is used to account for the case of imperfect demonstrations. [95] further extended the work of *APID* with a different evaluation loss:

$$\mathcal{L}^\pi = \mathbb{E}_{(s,a) \sim D_\pi} \| \mathcal{T}^* Q(s, a) - Q(s, a) \|, \tag{7}$$

where $\mathcal{T}^* Q(s, a) = R(s, a) + \gamma \mathbb{E}_{s' \sim p(.|s,a)} [\max_{a'} Q(s', a')]$. 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, [92] proposed the *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 method leverages the refined priority replay mechanism [99] where the probability of sampling a transition

i is based on its priority p_i with a temperature parameter α: $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$. Another algorithm named *LfDS* was proposed by [96], which draws a close connection to *reward shaping* (Section V-A). *LfDS* builds the potential value of a *state-action* pair as the highest similarity between the given pair and the expert demonstrations. This augmented reward assigns more credits to state-actions that are more similar to expert demonstrations, encouraging the agent for expert-like behavior.

Besides Q-learning, recent work has integrated *LfD* into ***policy gradient*** [63], [69], [78], [93], [97]. A representative work along this line is *Generative Adversarial Imitation Learning (GAIL)* [69]. *GAIL* introduced the notion of *occupancy measure* d_π, which is the stationary state-action distributions derived from a policy π. 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 π and the expert policy π_E. Specifically, the new reward is learned by adversarial training [62]: a discriminator D is learned to distinguish interactions sampled from the current policy π and the expert policy π_E:

$$J_D = \max_{D:\mathcal{S}\times\mathcal{A}\to(0,1)} \mathbb{E}_{d_\pi} \log[1 - D(s,a)] + \mathbb{E}_{d_E} \log[D(s,a)] \tag{8}$$

Since π_E is unknown, its state-action distribution d_E is estimated based on the given expert demonstrations 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 min-max optimization:

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

The philosophy in *GAIL* of using expert demonstrations for distribution matching has inspired other *LfD* algorithms. For example, [97] extended *GAIL* with an algorithm called *Policy Optimization from Demonstrations (POfD)*, which combines the discriminator reward with the environment reward:

$$\max_\theta = \mathbb{E}_{d_\pi}[r(s,a)] - \lambda D_{JS}[d_\pi || d_E]. \tag{9}$$

Both *GAIL* and *POfD* are under an *on-policy* RL framework. To further improve the sample efficiency of TL, some *off-policy* algorithms have been proposed, such as *DDPGfD* [78] 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 δ_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 ϵ 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 [93]. 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 [93] 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:

$$\mathcal{L}_{BC} = \sum_{i=1}^{|D_E|} \|\pi(s_i|\theta_\pi) - a_i\|^2 \mathbb{1}[Q(s_i,a_i) > Q(s_i, \pi(s_i))]. \tag{10}$$

There are several challenges faced by *LfD*, one of which is the *imperfect demonstrations*. Previous approaches usually presume near-oracle demonstrations. Towards tackling *suboptimal* demonstrations, [59] 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 regularized objective to alleviate overfitting on biased data [92], [99]. A different strategy is to leverage those sub-optimal demonstrations only to boost the initial learning stage. For instance, [63] proposed *Self-Adaptive Imitation Learning (SAIL)*, which learns from suboptimal 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* ([100]): 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 [101], decaying the effects of demonstration guidance by softening its regularization on policy learning over time [102], and introducing *disagreement regularizations* by training an ensemble of policies based on the given demonstrations, where the variance among policies serves as a negative reward function [103].

We summarize the above-discussed approaches in Table II. 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*). To validate the algorithm robustness given different knowledge resources, most *LfD* methods are evaluated using metrics that either indicate the performance under *limited* demonstrations (*nka*) or *suboptimal* demonstrations (*nka*). The integration of *LfD* with *off-policy* RL backbone makes it natural to adopt *pe* metrics for evaluating how learning efficiency can be further improved by knowledge transfer. Developing more general *LfD* approaches that are agnostic to RL frameworks and can learn from sub-optimal or limited demonstrations would be the ongoing focus for this research domain.

TABLE II
A COMPARISON OF LEARNING FROM DEMONSTRATION APPROACHES

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

C. Policy Transfer

Policy transfer is a TL approach where the external knowledge takes the form of pre-trained policies from one or multiple source domains. Work discussed in this section is built upon a *many-to-one* problem setting, described as below:

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

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

1) Transfer Learning Via Policy Distillation: The idea of *knowledge distillation* has been applied to the field of RL to enable *policy distillation*. Knowledge distillation was first proposed by [104] as an approach of knowledge ensemble from multiple teacher models into a single student model. Conventional policy distillation approaches transfer the teacher policy following a supervised learning paradigm [105], [106]. Specifically, a student policy is learned by minimizing the divergence of action distributions between the teacher policy π_E and student policy π_θ, which is denoted as $\mathcal{H}^{\times}(\pi_E(\tau_t)|\pi_\theta(\tau_t))$:

$$\min_{\theta} \mathbb{E}_{\tau\sim\pi_E}\left[\sum_{t=1}^{|\tau|} \nabla_\theta \mathcal{H}^{\times}(\pi_E(\tau_t)|\pi_\theta(\tau_t))\right]. \tag{11}$$

The above expectation is taken over trajectories sampled from the teacher policy π_E, hence this approach is called *teacher distillation*. One example along this line is [105], in which N teacher policies are learned for N source tasks separately, and each teacher yields a dataset $D^E = \{s_i, \boldsymbol{q}_i\}_{i=0}^{N}$ consisting of observations s and vectors of the corresponding Q-values $\boldsymbol{q}$, such that $\boldsymbol{q}_i = [Q(s_i,a_1), Q(s_i,a_2), \ldots | a_j \in \mathcal{A}]$. Teacher policies are further distilled to a single student π_θ by minimizing the KL-Divergence between each teacher $\pi_{E_i}(a|s)$ and the student π_θ, approximated using the dataset D^E: $\min_\theta \mathcal{D}_{KL}(\pi^E|\pi_\theta) \approx \sum_{i=1}^{|D^E|} \text{softmax}(\frac{\boldsymbol{q}_i^E}{\tau}) \ln(\frac{\text{softmax}(\boldsymbol{q}_i^E)}{\text{softmax}(\boldsymbol{q}_i^\theta)})$.

Another policy distillation approach is *student distillation* [51], [60], which is resemblant to teacher distillation except that during the optimization step, the objective expectation is taken over trajectories sampled from the student policy instead of the teacher policy, i.e.,: $\min_\theta \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=1}^{|\tau|} \nabla_\theta \mathcal{H}^{\times}(\pi_E(\tau_t)|\pi_\theta(\tau_t))]$. [60] summarized related work on both kinds of distillation approaches. Although it is feasible to combine both distillation approaches [100], we observe that more recent work focuses on student distillation, which empirically shows better exploration ability compared to teacher distillation, especially when the teacher policies are *deterministic*.

Taking an alternative perspective, there are two approaches of policy distillation: (1) minimizing the cross-entropy between the teacher and student policy distributions over actions [51], [107]; and (2) maximizing the probability that the teacher policy will visit trajectories generated by the student, i.e., $\max_\theta P(\tau \sim \pi_E|\tau \sim \pi_\theta)$ [50], [108]. One example of approach (1) is the *Actor-mimic* algorithm [51]. This algorithm distills the knowledge of expert agents into the student by minimizing the cross entropy between the student policy π_θ and each teacher policy π_{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. The teacher 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 [50]. which learns a *centroid* policy π_θ that is derived from K teacher policies. The knowledge in each teacher π_{E_i} is distilled to the centroid and get transferred to the student, while both the transition dynamics $\mathcal{T}_i$ and reward distributions $\mathcal{R}_i$ for source domain $\mathcal{M}_i$ are heterogeneous. The student policy is learned 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 π_θ are used as augmented rewards. Therefore, the above approach also draws a close

TABLE III
A COMPARISON OF POLICY TRANSFER APPROACHES

Paper	Transfer approach	MDP difference	RL framework	Evaluation metrics
[105]	Distillation	$\mathcal{S}, \mathcal{A}$	DQN	ap, ar
[106]	Distillation	$\mathcal{S}, \mathcal{A}$	DQN	ap, ar, pe, ps
[51]	Distillation	$\mathcal{S}, \mathcal{A}$	Soft Q-learning	ap, ar, tr, pe, ps
[50]	Distillation	$\mathcal{S}, \mathcal{A}$	A3C	ap, ar, pe, tt
[109]	Reuse	$\mathcal{R}$	Tabular Q-learning	ap, ar, ps, tr
[110]	Reuse	$\mathcal{R}$	DQN	ap, ar, pe, ps

connection to *Reward Shaping* (Section V-A). In effect, the $\log \pi_\theta(a_t|s_t)$ term guides the learning policy π_θ 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 [107] 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.

2) Transfer Learning Via Policy Reuse: Policy reuse directly reuses policies from source tasks to build the target policy. The notion of policy reuse was proposed by [109], which directly learns the target policy as a weighted combination of different source-domain policies, and 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 π_{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 [110] extended the *policy improvement* theorem [111] 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: $\left|Q^{\pi_i}(s,a) - \hat{Q}^{\pi_i}(s,a)\right| \leq \epsilon \forall s \in \mathcal{S}, a \in \mathcal{A}$, 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 [110], in which an expert policy π_{E_i} is also trained on a different source domain $\mathcal{M}_i$ with reward function $\mathcal{R}_i$, so that $Q^\pi_{\mathcal{M}_0}(s,a) \neq Q^\pi_{\mathcal{M}_i}(s,a)$. 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: $Q_i^\pi(s,a) = \psi(s,a)^T \mathbf{w}_i$. [110] 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 V-E2.

We summarize the abovementioned policy transfer approaches in Table III. In general, policy transfer can be realized by *knowledge distillation*, which can be either optimized from the student's perspecive (*student distillation*), or from the teacher's perspective (*teacher distillation*) Alternatively, teacher policies can also be directly *reused* to update the target policy. Regarding evaluation, most of the abovementioned work has investigated a multi-teacher transfer scenario, hence the *generalization* ability or *robustness* is largely evaluated on metrics such as *performance sensitivity(ps)* (e.g., performance given different numbers of teacher policies or source tasks). *Performance with fixed epochs (pe)* is another commonly shared metric to evaluate how the learned policy can quickly adapt to the target domain. All approaches discussed so far presumed one or multiple *expert* policies, which are always at the disposal of the learning agent. Open questions along this line include *How to leverage imperfect policies for knowledge transfer*, or *How to refer to teacher policies within a budget*.

D. Inter-Task Mapping

In this section, we review TL 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: 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* [66], [112]. One examples is [66] 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 $X_S(\mathcal{S}^t) \to \mathcal{S}^s$, $X_A(\mathcal{A}^t) \to \mathcal{A}^s$. Based on X_S and X_A, a mapping function over the Q-values $M(Q_s) \to Q_t$ can be derived accordingly. Another work is done by [112] 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 *learning* a mapping function [113], [114], [115]. 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 [113] and [114], the mapping function is learned on the *agent-specific* sub state, and the mapped representation is applied to reshape the immediate reward. For [113], 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 structure [116] to largely reserve information about the source domain. For transferring knowledge from the source agent to a new task, the environment reward is augmented with a shaped reward term to encourage the target agent to imitate the source agent on an embedded feature space:

$$r'(s,\cdot) = \alpha \left\| f(s^s_{agent};\theta_f) - g(s^t_{agent};\theta_g) \right\|, \qquad (12)$$

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

Another work is [115] which applied the *Unsupervised Manifold Alignment (UMA)* method [117] to automatically learn the state mapping. Their approach requires collecting trajectories from both the source and the target domain to learn such a mapping. While applying policy gradient learning, trajectories from the target domain $\mathcal{M}_t$ 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 $\widetilde{\tau}_s$, which are further mapped to the target domain: $\widetilde{\tau}_s \rightarrow \widetilde{\tau}_t$. The deviation between $\widetilde{\tau}_t$ and τ_t are used as a loss to be minimized in order to improve the target policy. Similar ideas of using *UMA* for inter-task mapping can also be found in [118] and [119].

In addition to approaches that utilizes mapping over states or actions, [120] 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 [121], 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 IV, for TL approaches that utilize an inter-task mapping, the mapped knowledge can be (a subset of) the state space [113], [114], the Q-function [66], or (representations of) the state-action-sate transitions [120]. In addition to being directly applicable in the target domain [120], the mapped representation can also be used as an augmented shaping reward [113], [114] or a loss objective [115] in order to guide the agent learning in the target domain. Most inter-task mapping methods tackle domains with moderate state-action space dimensions, such as maze tasks or tabular MDPs, where the goal can be reaching a target state with a minimal number of transitions. Accordingly, *tt* has been used to measure TL performance. For tasks with limited and discrete state-action space, evaluation is also conducted with different number of initial states collected in the target domain (*nka*).

E. Representation Transfer

This section review approaches that transfer knowledge in the form of representations learned by deep neural networks. They are built upon the following consensual assumption:

Assumption [Existence of Task-Invariance Subspace]: The state space ($\mathcal{S}$), action space ($\mathcal{A}$), or the reward space ($\mathcal{R}$) can be disentangled into orthogonal subspaces, which are task-invariant such that knowledge can be transferred between domains on the universal subspace.

We organize recent work along this line into two subtopics: 1) approaches that directly *reuse* representations from the source domain (Section V-E1), and 2) 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 V-E2).

1) Reusing Representations: A representative work of reusing representations is [122], 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*, and each column is a policy network for 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.

Progressive network comes with a cost of large network structures, as the network grows proportionally with the number of incoming tasks. A later framework called *PathNet* alleviates this issue by learning a network with a fixed size [123]. *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 [124].

TABLE IV
A COMPARISON OF INTER-TASK MAPPING APPROACHES

Methods	RL framework	MDP difference	Mapping function	Usage of mapping	Evaluation metrics
[66]	*SARSA*	$\mathcal{S}_t \neq \mathcal{S}_t, \mathcal{A}_s \neq \mathcal{A}_t$	$M(Q_s) \rightarrow Q_t$	Q value reuse	*ap, ar, tt, tr*
[112]	*Q-learning*	$\mathcal{A}_s \neq \mathcal{A}_t, \mathcal{R}_s \neq \mathcal{R}_t$	$M(Q_s) \rightarrow$ *advice*	Relative Q ranking	*ap, ar, tr*
[113]	–	$\mathcal{S}_s \neq \mathcal{S}_t$	$M(s_t) \rightarrow r'$	Reward shaping	*ap, ar, pe, tr*
[114]	*SARSA*(λ)	$\mathcal{S}_s \neq \mathcal{S}_t$ $\mathcal{R}_s \neq \mathcal{R}_t$	$M(s_t) \rightarrow r'$	Reward shaping	*ap, ar, pe, tt*
[115]	Fitted Value Iteration	$\mathcal{S}_s \neq \mathcal{S}_t$	$M(s_s) \rightarrow s_t$	Penalty loss on state deviation from expert policy	*ap, ar, pe, tr*
[121]	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	*ap, ar, pe, tr, nta*
[120]	–	$\mathcal{S}_s \times \mathcal{A}_s \neq \mathcal{S}_t \times \mathcal{A}_t$	$M((s_s, a_s, s'_s) \rightarrow (s_t, a_t, s'_t))$	Reduce random exploration	*ap, ar, pe, tr, nta*

"–" Indicates no RL framework constraints.

Another approach of reusing representations for TL is modular networks [52], [53], [125]. For example, [52] proposed to decompose the policy network into a task-specific module and agent-specific module. Specifically, let π be a policy performed by any agent (robot) r over the task $\mathcal{M}_k$ as a function ϕ 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 [125], 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 θ 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.

2) Disentangling Representations: Methods discussed in this section mostly focus on learning a *disentangled* representation. Specifically, we elaborate on TL 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: $\boldsymbol{\mathcal{M}} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_K\}$, 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 [126] 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 function, *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, $\boldsymbol{\psi}$ and R: $V^\pi(s) = \sum_{s'} \boldsymbol{\psi}(s, s')\mathbf{w}(s')$, where $\mathbf{w}(s')$ is a reward mapping function that maps states to scalar rewards, and $\boldsymbol{\psi}$ is the *SR* which describes any state s as the occupancy measure of the future occurred states when following π, with $\mathbb{1}[S = s'] = 1$ as an indicator function:

$$\boldsymbol{\psi}(s, s') = \mathbb{E}_\pi \left[\sum_{i=t}^{\infty} \gamma^{i-t} \mathbb{1}[S_i = s'] | S_t = s \right].$$

The *successor* nature of *SR* makes it learnable using any TD-learning algorithms. Especially, [126] 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 [110] 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* $\boldsymbol{\psi}^\pi(s, a)$ and the *reward mapper* $\mathbf{w}$; (iii) Generalized policy improvement (GPI) algorithm is introduced to accelerate policy transfer for multi-tasks (Section V-C2). 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 π in a task. The advantage of *SR* is that, when the knowledge of $\boldsymbol{\psi}^\pi(s, a)$ in the source domain $\mathcal{M}_s$ is observed, one can quickly get the performance evaluation of the same policy in the target domain $\mathcal{M}_t$ by replacing $\mathbf{w}_s$ with $\mathbf{w}_t$: $Q^\pi_{\mathcal{M}_t} = \boldsymbol{\psi}^\pi(s, a)\mathbf{w}_t$. Similar ideas of learning *SR* as a TD-algorithm on a latent representation $\phi(s, a, s')$ can also be found in [127], [128]. Specifically, the work of [127] 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 [67] 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, [67] aims to learn a matrix $\boldsymbol{\phi}(s,a,s') \in \mathbb{R}^{D\times d}$ 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 $\boldsymbol{\phi}(s,a,s')$ 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:

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

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

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

A similar idea of using reward functions as features to represent unseen tasks is also proposed by [129], which assumes the $\boldsymbol{\psi}$ and $\mathbf{w}$ as observable quantities from the environment.

Universal Function Approximation **(UVFA)** is an alternative approach of learning disentangled state representations [64]. Same as SR, *UVFA* allows TL 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 of *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 $\boldsymbol{\psi}(g)$, 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.

UVFA has been connected to *SR* by [67], 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 [130]. Following the same linearity assumption, *USFA* is proposed as a function over a triplet of the state, action, and a policy embedding z: $\phi(s,a,z) : \mathcal{S}\times\mathcal{A}\times\mathbb{R}^k \to \mathbb{R}^d$, where z is the output of a *policy-encoding mapping* $z = e(\pi) : \mathcal{S}\times\mathcal{A} \to \mathbb{R}^k$. Based on *USFA*, the Q-function of any policy π for a task specified by $\mathbf{w}$ can be formularized as the product of a reward-agnostic *Universal Successor Feature (USF)* $\boldsymbol{\psi}$ and a reward mapper $\mathbf{w}$: $Q(s,a,\mathbf{w},z) = \boldsymbol{\psi}(s,a,z)^\top \mathbf{w}$. Facilitated by the disentangled rewards and policy generalization, [130] further introduced a generalized TD-error as a function over tasks $\mathbf{w}$ and policy z, which allows them to approximate the Q-function of any policy on any task using a TD-algorithm.

3) Summary and Discussion: We provide a summary of the discussed work in this section in Table V. Representation transfer can facilitate TL in multiple ways based on assumptions about certain task-invariant property. Some assume that tasks are different only in terms of their reward distributions. Other stronger assumptions include (i) decoupled dynamics, rewards [110], or policies [130] from the Q-function representations, and (ii) the feasibility of defining tasks in terms of states [130]. Based on those assumptions, approaches such as TD-algorithms [67] or matrix-factorization [64] become applicable to learn such disentangled representations. To further exploit the effectiveness of disentangled structure, we consider that *generalization* approaches, which allow changing dynamics or state distributions, are important future work that is worth more attention in this domain.

Most discussed work in this section tackles multi-task RL or meta-RL scenarios, hence the agent's *generalization* ability is extensively investigated. For instance, methods of modular networks largely evaluated the *zero-shot* performance from the meta-RL perspective [52], [130]. Given a fixed number of training epochs (*pe*), *Transfer ratio (tr)* is manifested differently among these methods. It can be the relative performance of a modular net architecture compared with a baseline, or the accumulated return in modified target domains, where reward scores are negated for evaluating the dynamics transfer. *Performance sensitivity (ps)* is also broadly studied to estimate the *robustness* of TL. [110] analyzed the performance sensitivity given varying source tasks, while [130] studied the performance on different unseen target domains.

There are unresolved questions in this intriguing research topic. One is *how to handle drastic changes of reward functions between domains*. As discussed in [131], 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 [67] to represent unseen tasks (reward functions), or decoupling policies from Q-function representation [130] 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.

Another question is *how to generalize the representation learning for TL 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

TABLE V
A COMPARISON OF TL APPROACHES OF REPRESENTATION TRANSFER

Methods	Representations format	Assumptions	MDP difference	Learner	Evaluation metrics
Progressive Net [122]	Lateral connections to previously learned network modules	N/A	$\mathcal{S}, \mathcal{A}$	A3C	ap, ar, pe, ps, tr
PathNet [123]	Selected neural paths	N/A	$\mathcal{S}, \mathcal{A}$	A3C	ap, ar, pe, tr
Modular Net [52]	Task(agent)-specific network module	Disentangled state representation	$\mathcal{S}, \mathcal{A}$	Policy Gradient	ap, ar, pe, tt
Modular Net [125]	Dynamic transitions module learned on state latent representations.	N/A	$\mathcal{S}, \mathcal{A}$	A3C	ap, ar, pe, tr, ps
SR [110]	*SF*	Reward function can be linearly decoupled	$\mathcal{R}$	DQN	ap, ar, nka, ps
SR [127]	Encoder-decoder learned *SF*	N/A	$\mathcal{R}$	DQN	ap, ar, pe, ps
SR [67]	Encoder-decoder learned *SF*	Rewards can be represented by set of basis functions	$\mathcal{R}$	$Q(\lambda)$	ap, pe
UVFA [64]	Matrix-factorized UF	Goal conditional RL	$\mathcal{R}$	Tabular Q-learning	ap, ar, pe, ps
UVFA with SR [130]	Policy-encoded UF	Reward function can be linearly decoupled	$\mathcal{R}$	ϵ-greedy Q-learning	ap, ar, pe

of occupancy measure for SR 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 [132]. Alternatively, TL mechanisms from the supervised learning domain, such as meta-learning, which enables the ability of fast adaptation to new tasks [133], or importance sampling [134], which can compensate for the prior distribution changes [10], may also shed light on this question.

VI. APPLICATIONS

In this section we summarize recent applications that are closely related to using TL techniques for tackling RL domains.

Robotics Learning is a prominent application domain of RL. TL approaches in this field include *robotics learning from demonstrations*, where expert demonstrations from humans or other robots are leveraged [135] Another is *collaborative robotic training* [136], [137], in which knowledge from different robots is transferred by sharing their policies and episodic demonstrations. Recent research focus is this domain is fast and robust adaptation to unseen tasks. One example towards this goal is [138], in which robust robotics policies are trained using synthetic demonstrations to handle dynamic environments. Another solution is to learn domain-invariant latent representations. Examples include [139], which learns the latent representation using 3D CAD models, and [140], [141] which are derived based on the Generative-Adversarial Network. Another example is *DARLA* [142], which is a zero-shot transfer approach to learn disentangled representations that are robust against domain shifts. We refer readers to [70], [143] for detailed surveys along this direction.

Game Playing is a common test-bed for TL and RL algorithms. It has evolved from classical benchmarks such as grid-world games to more complex settings such as online-strategy games or video games with multimodal inputs. One example is *AlphaGo*, which is an algorithm for learning the online chessboard games using both TL and RL techniques [90]. *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* [144], even beat the world's first ranked human player. TL-DRL approaches are also thriving in video game playing. Especially, OpenAI has trained *Dota2* agents that can surpass human experts [145]. State-of-the-art platforms include *MineCraft*, *Atari*, and *Starcraft*. [146] designed new RL benchmarks under the *MineCraft* platform. [147] provided a comprehensive survey on DL applications in video game playing, which also covers TL and RL strategies from certain perspectives. A large portion of TL approaches reviewed in this survey have been applied to the *Atari* platforms [148].

Natural Language Processing (NLP) has evolved rapidly along with the advancement of DL and RL. Applications of RL to NLP range widely, from *Question Answering (QA)* [149], *Dialogue systems* [150], *Machine Translation* [151], to an integration of NLP and Computer Vision tasks, such as *Visual Question Answering (VQA)* [152], *Image Caption* [153], etc. Many NLP applications have implicitly applied TL approaches. Examples include learning from expert demonstrations for *Spoken Dialogue Systems* [154], *VQA* [152]; or reward shaping for *Sequence Generation* [155], *Spoken Dialogue Systems* [80],*QA* [81], [156], and *Image Caption* [153], or transferring policies for *Structured Prediction* [157] and *VQA* [158].

Large Model Training: RL from human and model-assisted feedback becomes indispensable in training large models (LMM), such as GPT4 [159], Sparrow [160], PaLM [161], LaMDA [162], which have shown tremendous breakthrough in dialogue applications, search engine answer improvement, artwork generation, etc. The TL method at the core of them is using human preferences as a reward signal for model fine-tuning, where the preference ranking itself is considered as shaped rewards. We believe that TL with carefully crafted human knowledge will help better align large models with human intent and hence achieve trustworthy and de-biased AI.

Health Informatics: RL has been applied to various healthcare tasks [163], including *automatic medical diagnosis* [164], [165], *health resource scheduling* [166], and *drug discovery and development*, [167], [168], etc. Among these applications we observe emerging trends of leveraging prior knowledge to improve the RL procedure, especially given the difficulty of accessing large amounts of clinical data. Specifically, [169] 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. [170] 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. [171] studied applying human-involved interactive RL training for health informatics.

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], [172], [173]. We refer readers to [174] for a review along this line. Deep RL are also effective solutions to problems in ***Finance***, including *portfolio management* [175], [176], *asset allocation* [177], and *trading optimization* [178]. Another application is the ***Electricity Systems***, especially the *intelligent electricity networks*, which can benefit from RL techniques for improved power-delivery decisions [179], [180] and active resource management [181]. [7] provides a detailed survey of RL techniques for electric power system applications.

VII. Future Perspectives

In this section, we present some open challenges and future directions in TL that are closely related to the DRL domain, based on both retrospectives of the methods discussed in this survey and outlooks to the emerging trends of AI.

Transfer Learning From Black-Box: Ranging from exterior teacher demonstrations to pre-trained function approximators, black-box resource is more accessible and predominant than well-articulated knowledge. Therefore, leveraging such black-box resource is indispensable for practical TL in DRL. One of its main challenges resides in ***estimating the optimality*** of black-box resource, which can be potentially noisy or biased. We consider that efforts can be made from the following perspectives:

1) Inferring the *reasoning* mechanism inside the black-box. Resemblant ideas have been explored in *inverse RL* and *model-based RL*, where the goal is to approximate the reward function or to learn the dynamics model under which the demonstrated knowledge becomes reasonable.
2) Designing effective *feedback* schemes, including leveraging domain-provided rewards, intrinsic reward feedback, or using human preference as feedback.
3) Improving the *interpretability* of the transferred knowledge [182], [183], which benefits in evaluating and explaining the process of TL from black-box. It can also alleviate catastrophic decision-making for high-stake tasks such as auto-driving.

Knowledge Disentanglement and Fusion are both towards better knowledge sharing across domains. Disentangling knowledge is usually a *prerequisite* for efficient knowledge fusion, which may involve external knowledge from multiple source *domains*, with diverging *qualities*, presented in different *modalities*, etc. Disentangling knowledge in RL can be interpreted from different perspectives: i) disentangling the action, state, or reward representations, as discussed in Sec V-E; 2) decomposing complex tasks into multiple *skill snippets*. The former is an effective direction in tackling meta-RL and multi-task RL, although some solutions hinge on strict assumptions of the problem setting, such as linear dependence among domain dynamics or learning goals. The latter is relevant to hierarchical RL and *prototype learning* from sequence data [184]. It is relatively less discussed besides few pioneer research [132]. We believe that this direction is worth more research efforts, which not only benefits interpretable knowledge learning, but also aligns with human perception.

Framework-Agnostic Knowledge Transfer: Most contemporary TL approaches are designed for certain RL frameworks. Some are applicable to RL algorithms designed for the discrete-action space, while others may only be feasible given a continuous action space. One fundamental reason behind is the diversified development of RL algorithms. We expect that unified RL frameworks would contribute to the standardization of TL approaches in this field.

Evaluation and Benchmarking: Variant evaluation metrics have been proposed to measure TL from different but complementary perspectives, although no single metric can summarize the efficacy of a TL approach. Designing a set of generalized, novel metrics is beneficial for the development of TL in DRL. Moreover, with the effervescent development of large-scale models, it is crucial to standardize evaluation from the perspectives of *ethics* and *groundedness*. The *appropriateness* of the transferred knowledge, such as potential *stereotypes* in human preference, and the *bias* in the model itself should also be quantified as metrics.

Knowledge Transfer to and From Pre-Trained Large Models: By the time of this survey being finalized, unprecedented DL breakthroughs have been achieved in learning large-scale models built on massive computation resource and attributed data. One representative example is the Generative Pre-trained Transformer (GPT) [159]. Considering them as complete ***knowledge graphs*** whose training process maybe inaccessible, there are more challenges in this direction besides learning from a *black-box*, which are faced by a larger AI community including the RL domain. We briefly point out two directions that are worth ongoing attention:

1) *Efficient model fine-tuning with knowledge distillation:* One important method for fine-tuning large models is *RL with human feedback*, in which the quantity and quality of human ratings are critical for realizing a good reward model. We anticipate other forms of TL methods in RL to be explored to further improve the efficiency of fine-tuning, such as imitation learning with adversarial training to achieve human-level performance.
2) *Principled prompt-engineering for knowledge extraction:* More often the large model itself cannot be accessed,

but only input and output of models are allowed. Such inference based knowledge extraction requires delicate prompt designs. Some efficacious efforts include designing prompts with mini task examples as one-shot learning, *decomposing* complex tasks into *architectural*, contextual prompts. Prompt engineering is being proved an important direction for effective knowledge extraction, which with proper design, can largely benefit downstream tasks that depend on large model resources.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," 2017, *arXiv: 1708.05866*.

[3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, pp. 1334–1373, 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," *Int. J. Robot. Res.*, vol. 37, pp. 421–436, 2018.

[5] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, 2013.

[6] M. R. Kosorok and E. E. Moodie, *Adaptive TreatmentStrategies in Practice: Planning Trials and Analyzing Data for Personalized Medicine*. Philadelphia, PA, USA: 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*, vol. 50, pp. 6918–6927, 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 Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1140–1150, Sep. 2013.

[9] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2496–2505.

[10] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2009.

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

[12] A. Lazaric, *Transfer in Reinforcement Learning: A Framework and a Survey*. Berlin, Germany: Springer, 2012.

[13] R. Bellman, "A Markovian decision process," *J. Math. Mechanics*, vol. 1957, pp. 679–684, 1957.

[14] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 449–458.

[15] M. Liu, M. Zhu, and W. Zhang, "Goal-conditioned reinforcement learning: Problems and solutions," 2022, *arXiv:2201.08299*.

[16] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 1515–1528.

[17] Z. Xu and A. Tewari, "Reinforcement learning in factored MDPs: Oracle-efficient algorithms and tighter regret bounds for the non-episodic setting," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 18 226–18 236.

[18] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 24 611–24 624.

[19] 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," 2018, *arXiv: 1809.02925*.

[20] G. A. Rummery and M. Niranjan, *On-Line Q-Learning Using Connectionist Systems*. Cambridge, U.K.: Univ. Cambridge, Department of Engineering Cambridge, 1994.

[21] H. Van Seijen, H. Van Hasselt, S. Whiteson, and M. Wiering, "A theoretical and empirical analysis of Expected Sarsa," in *Proc. IEEE Symp. Adaptive Dynamic Program. Reinforcement Learn.*, 2009, pp. 177–184.

[22] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.

[23] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[24] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[25] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, 1992.

[26] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[27] M. Hessel, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2018, pp. 3215–3222.

[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992.

[29] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv: 1707.06347*.

[31] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

[32] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[33] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, *arXiv: 1802.09477*.

[34] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7559–7566.

[35] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "The cross-entropy method for optimization," in *Handbook of Statistics*, vol. 31. Amsterdam, The Netherlands: Elsevier, 2013, pp. 35–59.

[36] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4759–4770.

[37] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proc. Conf. Mach. Learn.*, Elsevier, 1990, pp. 216–224.

[38] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, "Model-based value estimation for efficient model-free reinforcement learning," 2018, *arXiv: 1803.00101*.

[39] S. Levine and V. Koltun, "Guided policy search," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2013, pp. 1–9.

[40] H. Bharadhwaj, K. Xie, and F. Shkurti, "Model-predictive control via cross-entropy and gradient-based optimization," in *Proc. Learn. Dyn. Control*, PMLR, 2020, pp. 277–286.

[41] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 465–472.

[42] Y. Gal, R. McAllister, and C. E. Rasmussen, "Improving PILCO with Bayesian neural network dynamics models," in *Proc. Data-efficient Mach. Learn. Workshop*, 2016, Art. no. 25.

[43] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 951–958.

[44] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1993, pp. 271–278.

[45] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, pp. 181–211, 1999.

[46] R. Parr and S. J. Russell, "Reinforcement learning with hierarchies of machines," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1998, pp. 1043–1049.

[47] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *J. Artif. Intell. Res.*, vol. 13, pp. 227–303, 2000.

[48] A. Lazaric and M. Ghavamzadeh, "Bayesian multi-task reinforcement learning," in *Proc. 27th Int. Conf. Mach. Learn.*, Omnipress, 2010, pp. 599–606.

[49] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 12, pp. 5586–5609, Dec. 2022.

[50] Y. Teh et al., "Distral: Robust multitask reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4499–4509.

[51] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2016.

[52] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 2169–2176.

[53] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 166–175.

[54] R. Yang, H. Xu, Y. Wu, and X. Wang, "Multi-task reinforcement learning with soft modularization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 4767–4777.

[55] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5149–5169, Sep. 2022.

[56] Z. Jia, X. Li, Z. Ling, S. Liu, Y. Wu, and H. Su, "Improving policy optimization with generalist-specialist learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2022, pp. 10 104–10 119.

[57] W. Ding, H. Lin, B. Li, and D. Zhao, "Generalizing goal-conditioned reinforcement learning with variational causal reasoning," 2022, *arXiv:2207.09081*.

[58] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A survey of zero-shot generalisation in deep reinforcement learning," *J. Artif. Intell. Res.*, vol. 76, pp. 201–264, 2023.

[59] B. Kim, A.-M. Farahmand, J. Pineau, and D. Precup, "Learning from limited demonstrations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2859–2867.

[60] W. Czarnecki et al., "Distilling policy distillation," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 1331–1340.

[61] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 278–287.

[62] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[63] Z. Zhu, K. Lin, B. Dai, and J. Zhou, "Learning sparse rewarded tasks from sub-optimal demonstrations," 2020, *arXiv: 2004.00530*.

[64] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1312–1320.

[65] C. Finn and S. Levine, "Meta-learning: From few-shot learning to rapid reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019.

[66] M. E. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning," *J. Mach. Learn. Res.*, vol. 8, pp. 2125–2167, 2007.

[67] A. Barreto et al., "Transfer in deep reinforcement learning using successor features and generalised policy improvement," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 501–510.

[68] Z. Zhu, K. Lin, B. Dai, and J. Zhou, "Off-policy imitation learning from observations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 12402–12413.

[69] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4572–4580.

[70] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2020, pp. 737–744.

[71] M. Muller-Brockhausen, M. Preuss, and A. Plaat, "Procedural content generation: Better benchmarks for transfer reinforcement learning," in *Proc. IEEE Conf. Games*, 2021, pp. 01–08.

[72] N. Vithayathil Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics*, vol. 9, no. 9, 2020, Art. no. 1363.

[73] R. J. Williams and L. C. Baird, "Tight performance bounds on greedy policies based on imperfect value functions," Northeastern Univ., College Comput. Sci., Boston, MA, Tech. Rep. NU-CCS-93-14, 1993.

[74] E. Wiewiora, G. W. Cottrell, and C. Elkan, "Principled methods for advising reinforcement learning agents," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 792–799.

[75] S. M. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2012, pp. 433–440.

[76] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowé, "Expressing arbitrary reward functions as potential-based advice," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2015, pp. 2652–2658.

[77] T. Brys, A. Harutyunyan, M. E. Taylor, and A. Nowé, "Policy transfer using reward shaping," in *Proc. Int. Conf. Anal. Appl. Math.*, 2015, pp. 181–188.

[78] M. Večerík et al., "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2017, *arXiv: 1707.08817*.

[79] A. C. Tenorio-Gonzalez, E. F. Morales, and L. Villasenor-Pineda, "Dynamic reward shaping: Training a robot by voice," in *Proc. Adv. Artif. Intell.*, 2010, pp. 483–492.

[80] 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," 2015, *arXiv:1508.03391*.

[81] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," 2018, *arXiv: 1808.10568*.

[82] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer, "Potential-based difference rewards for multiagent reinforcement learning," in *Proc. Int. Conf. Anal. Appl. Math.*, 2014, pp. 165–172.

[83] M. Grzes and D. Kudenko, "Learning shaping rewards in model-based reinforcement learning," in *Proc. AAMAS Workshop Adaptive Learn. Agents*, 2009, Art. no. 30.

[84] O. Marom and B. Rosman, "Belief reward shaping in reinforcement learning," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2018, pp. 3762–3769.

[85] F. Liu, Z. Ling, T. Mu, and H. Su, "State alignment-based imitation learning," 2019, *arXiv: 1911.10947*.

[86] K. Kim, Y. Gu, J. Song, S. Zhao, and S. Ermon, "Domain adaptive imitation learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5286–5295.

[87] Y. Ma, Y.-X. Wang, and B. Narayanaswamy, "Imitation-regularized offline learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2019, pp. 2956–2965.

[88] M. Yang and O. Nachum, "Representation matters: Offline pretraining for sequential decision making," 2021, *arXiv:2102.05815*.

[89] X. Zhang and H. Ma, "Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations," 2018, *arXiv: 1801.10459*.

[90] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016.

[91] S. Schaal, "Learning from demonstration," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1997, pp. 1040–1046.

[92] T. Hester et al., "Deep Q-learning from demonstrations," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2018, pp. 3223–3230.

[93] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6292–6299.

[94] J. Chemali and A. Lazaric, "Direct policy iteration with demonstrations," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 3380–3386.

[95] B. Piot, M. Geist, and O. Pietquin, "Boosted Bellman residual minimization handling expert demonstrations," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2014, pp. 549–564.

[96] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 3352–3358.

[97] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2469–2478.

[98] D. P. Bertsekas, "Approximate policy iteration: A survey and some new methods," *J. Control Theory Appl.*, vol. 9, pp. 310–335, 2011.

[99] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Representations*, 2016.

[100] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 627–635.

[101] Y. Gao et al., "Reinforcement learning from imperfect demonstrations," 2018, *arXiv: 1802.05313*.

[102] M. Jing et al., "Reinforcement learning from imperfect demonstrations under soft expert guidance," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2020, pp. 5109–5116.

[103] K. Brantley, W. Sun, and M. Henaff, "Disagreement-regularized imitation learning," in *Proc. Int. Conf. Learn. Representations*, 2019.

[104] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. Deep Learn. Representation Learn. Workshop*, 2014.

[105] A. A. Rusu, "Policy distillation," 2015, *arXiv:1511.06295*.

[106] H. Yin and S. J. Pan, "Knowledge transfer for deep reinforcement learning with hierarchical experience replay," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2017, pp. 1640–1646.

[107] S. Schmitt et al., "Kickstarting deep reinforcement learning," 2018, *arXiv: 1803.03835*.

[108] J. Schulman, X. Chen, and P. Abbeel, "Equivalence between policy gradients and soft Q-learning," 2017, *arXiv: 1704.06440*.
[109] F. Fernández and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in *Proc. 5th Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2006, pp. 720–727.
[110] A. Barreto et al., "Successor features for transfer in reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4058–4068.
[111] R. Bellman, "Dynamic programming," *Science*, vol. 153, pp. 34–37, 1966.
[112] L. Torrey, T. Walker, J. Shavlik, and R. Maclin, "Using advice to transfer knowledge acquired in one reinforcement learning task to another," in *Proc. Eur. Conf. Mach. Learn.*, 2005, pp. 412–424.
[113] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.
[114] G. Konidaris and A. Barto, "Autonomous shaping: Knowledge transfer in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2006, pp. 489–496.
[115] H. B. Ammar and M. E. Taylor, "Reinforcement learning transfer via common subspaces," in *Proc. 11th Int. Conf. Adaptive Learn. Agents*, 2012, pp. 21–36.
[116] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
[117] C. Wang and S. Mahadevan, "Manifold alignment without correspondence," in *Proc. Int. Joint Conf. on Artif. Intell.*, 2009, Art. no. 3.
[118] B. Bocsi, L. Csató, and J. Peters, "Alignment-based transfer learning for robot models," in *Proc. Int. Joint Conf. Neural Netw.*, 2013, pp. 1–7.
[119] H. B. Ammar, E. Eaton, P. Ruvolo, and M. E. Taylor, "Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2015, pp. 2504–2510.
[120] H. B. Ammar, K. Tuyls, M. E. Taylor, K. Driessens, and G. Weiss, "Reinforcement learning transfer via sparse coding," in *Proc. Int. Conf. Anal. Appl. Math.*, 2012, pp. 383–390.
[121] A. Lazaric, M. Restelli, and A. Bonarini, "Transfer of samples in batch reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 544–551.
[122] A. A. Rusu et al., "Progressive neural networks," 2016, *arXiv:1606.04671*.
[123] C. Fernando, "Pathnet: Evolution channels gradient descent in super neural networks," 2017, *arXiv: 1701.08734*.
[124] I. Harvey, "The microbial genetic algorithm," in *Proc. Eur. Conf. Artif. Life*, 2009, pp. 126–133.
[125] A. Zhang, H. Satija, and J. Pineau, "Decoupling dynamics and reward for transfer learning," 2018, *arXiv: 1804.10689*.
[126] P. Dayan, "Improving generalization for temporal difference learning: The successor representation," *Neural Comput.*, vol. 5, no. 4, pp. 613–624, Jul. 1993.
[127] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman, "Deep successor reinforcement learning," 2016, *arXiv:1606.02396*.
[128] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2371–2378.
[129] N. Mehta, S. Natarajan, P. Tadepalli, and A. Fern, "Transfer in variable-reward hierarchical reinforcement learning," *Mach. Learn.*, vol. 73, pp. 289–312, 2008.
[130] D. Borsa et al., "Universal successor features approximators," in *Proc. Int. Conf. Learn. Representations*, 2019.
[131] L. Lehnert, S. Tellex, and M. L. Littman, "Advantages and limitations of using successor features for transfer in reinforcement learning," 2017, *arXiv: 1708.00102*.
[132] J. C. Petangoda, S. Pascual-Diaz, V. Adam, P. Vrancx, and J. Grau-Moya, "Disentangled skill embeddings for reinforcement learning," 2019, *arXiv: 1906.09223*.
[133] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
[134] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *Proc. Int. Conf. Mach. Learn.*, 2004, Art. no. 114.
[135] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, pp. 469–483, 2009.
[136] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. Automat. Sci. Eng.*, vol. 12, no. 2, pp. 398–409, Apr. 2015.
[137] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3389–3396.
[138] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," 2017, *arXiv: 1702.02453*.
[139] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," 2016, *arXiv:1611.04201*.
[140] K. Bousmalis et al., "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4243–4250.
[141] H. Bharadhwaj, Z. Wang, Y. Bengio, and L. Paull, "A data-efficient framework for training and sim-to-real transfer of navigation policies," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 782–788.
[142] I. Higgins et al., "DARLA: Improving zero-shot transfer in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1480–1490.
[143] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, pp. 1238–1274, 2013.
[144] D. Silver et al., "Mastering the game of Go without human knowledge," *Nature*, vol. 550, pp. 354–359, 2017.
[145] OpenAI, Dotal2 blog, 2019. [Online]. Available: https://openai.com/blog/openai-five/
[146] J. Oh, V. Chockalingam, S. Singh, and H. Lee, "Control of memory, active perception, and action in Minecraft," 2016, *arXiv:1605.09128*.
[147] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Trans. Games*, vol. 12, no. 1, pp. 1–20, Mar. 2020.
[148] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[149] H. Chen, X. Liu, D. Yin, and J. Tang, "A survey on dialogue systems: Recent advances and new frontiers," *ACM SIGKDD Explorations Newslett.*, vol. 19, pp. 25–35, 2017.
[150] S. P. Singh, M. J. Kearns, D. J. Litman, and M. A. Walker, "Reinforcement learning for spoken dialogue systems," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2000, pp. 956–962.
[151] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.
[152] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko, "Learning to reason: End-to-end module networks for visual question answering," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 804–813.
[153] Z. Ren, X. Wang, N. Zhang, X. Lv, and L.-J. Li, "Deep reinforcement learning-based image captioning with embedding reward," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1151–1159.
[154] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Learning to compose neural networks for question answering," 2016, *arXiv:1601.01705*.
[155] D. Bahdanau et al., "An actor-critic algorithm for sequence prediction," 2016, *arXiv:1607.07086*.
[156] F. Godin, A. Kumar, and A. Mittal, "Learning when not to answer: A ternary reward structure for reinforcement learning based question answering," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, 2019, pp. 122–129.
[157] K.-W. Chang, A. Krishnamurthy, A. Agarwal, J. Langford, and H. Daumé III, "Learning to search better than your teacher," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2058–2066.
[158] 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," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 313–323.
[159] "GPT-4 technical report," 2023, *arXiv:2303.08774*.
[160] A. Glaese et al., "Improving alignment of dialogue agents via targeted human judgements," 2022, *arXiv:2209.14375*.
[161] A. Chowdhery et al., "PaLM: Scaling language modeling with pathways," 2022, *arXiv:2204.02311*.
[162] R. Thoppilan et al., "LaMDA: Language models for dialog applications," 2022, *arXiv:2201.08239*.
[163] C. Yu, J. Liu, and S. Nemati, "Reinforcement learning in healthcare: A survey," 2019, *arXiv: 1908.08796*.
[164] A. Alansary et al., "Evaluating reinforcement learning agents for anatomical landmark detection," *Med. Image Anal.*, vol. 53, pp. 156–164, 2019.
[165] K. Ma et al., "Multimodal image registration with deep context reinforcement learning," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervention*, 2017, pp. 240–248.

[166] T. T. Gomes, "Reinforcement learning for primary care e appointment scheduling," 2017.
[167] 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," in *Proc. Int. Conf. Parallel Process. Companion*, 2018, pp. 1–8.
[168] M. Popova, O. Isayev, and A. Tropsha, "Deep reinforcement learning for de novo drug design," *Sci. Adv.*, vol. 4, 2018, Art. no. eaap7885.
[169] 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," in *Proc. 4th Int. Conf. Mach. Learn. Appl.*, 2005, pp. 207–214.
[170] T. W. Killian, S. Daulton, G. Konidaris, and F. Doshi-Velez, "Robust and efficient transfer learning with hidden parameter Markov decision processes," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6251–6262.
[171] A. Holzinger, "Interactive machine learning for health informatics: When do we need the human-in-the-loop?," *Brain Inform.*, vol. 3, pp. 119–131, 2016.
[172] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA J. Automatica Sinica*, vol. 3, no. 3, pp. 247–254, Jul. 2016.
[173] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1774–1783.
[174] 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 Comput. Surv.*, vol. 50, pp. 1–38, 2017.
[175] J. Moody, L. Wu, Y. Liao, and M. Saffell, "Performance functions and reinforcement learning for trading systems and portfolios," *J. Forecasting*, vol. 17, pp. 441–470, 1998.
[176] Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning," in *Proc. IEEE Intell. Syst. Conf.*, 2017, pp. 905–913.
[177] R. Neuneier, "Enhancing Q-learning for optimal asset allocation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1998, pp. 936–942.
[178] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, Mar. 2017.
[179] G. Dalal, E. Gilboa, and S. Mannor, "Hierarchical decision making in electricity grid management," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2197–2206.
[180] 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 Trans. Smart Grid*, vol. 8, no. 5, pp. 2149–2159, Sep. 2017.
[181] Z. Wen, D. O'Neill, and H. Maei, "Optimal demand response using device-based reinforcement learning," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2312–2324, Sep. 2015.
[182] Y. Li, J. Song, and S. Ermon, "InfoGAIL: Interpretable imitation learning from visual demonstrations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3815–3825.
[183] R. Ramakrishnan and J. Shah, "Towards interpretable explanations for transfer learning in sequential tasks," in *Proc. AAAI Spring Symp. Ser.*, 2016.
[184] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart, "RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3512–3520.

Zhuangdi Zhu received the PhD degree from the Computer Science department of Michigan State University. She is currently a senior data and applied scientist with Microsoft. She has regularly published on prestigious machine learning conferences including NeurIPs, ICML, KDD, AAAI, etc. Her research interests reside in both fundamental and applied machine learning. Her current research involves reinforcement learning and distributed machine learning.

Kaixiang Lin received the PhD degree from Michigan State University. He is an applied scientist with Amazon web services. 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 with the Department of Computer Science and Engineering, Michigan State University. His research interests include pattern recognition and biometric authentication. He served as the editor-in-chief of *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 (Member, IEEE) received the PhD degree in computer science from Arizona State University in 2014. He is an associate professor with the Department of Computer Science and Engineering, Michigan State University. 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.