The Knowledge Engineering Review, Vol. 35, e7, 1 of 16. © Cambridge University Press, 2020
doi:10.1017/S0269888920000107

Domain adaptation-based transfer learning using adversarial networks

FARZANEH SHOELEH[1], MOHAMMAD MEHDI YADOLLAHI[1], and MASOUD ASADPOUR[2]

[1]*University of New Brunswick, Fredericton, New Brunswick, Canada*
e-mails: fshoeleh@unb.ca, mehdiyadollahi@unb.ca
[2]*University of Tehran, Tehran, Iran*
e-mail: asadpour@ut.ac.ir

Abstract

There is an implicit assumption in machine learning techniques that each new task has no relation to the tasks previously learned. Therefore, tasks are often addressed independently. However, in some domains, particularly reinforcement learning (RL), this assumption is often incorrect because tasks in the same or similar domain tend to be related. In other words, even though tasks are quite different in their specifics, they may have general similarities, such as shared skills, making them related. In this paper, a novel domain adaptation-based method using adversarial networks is proposed to do transfer learning in RL problems. Our proposed method incorporates skills previously learned from source task to speed up learning on a new target task by providing generalization not only within a task but also across different, but related tasks. The experimental results indicate the effectiveness of our method in dealing with RL problems.

1 Introduction

The reinforcement learning (RL) paradigm is a popular way for an autonomous agent to learn from experience with minimal feedback. The required learning time and the curse of dimensionality restrict the applicability of RL on real-world problems. According to the literature (Sutton *et al.*, 1999; Konidaris and Barto, 2009; Moradi *et al.*, 2012), it is believed that state abstraction methods and hierarchical architectures can improve the learning curve and lessen the hampering effect of the curse of dimensionality. While significant progress has been made to improve learning in a single task, the idea of transfer learning (TL) has been applied to RL tasks recently; (Barreto *et al.*, 2017; Shoeleh & Asadpour, 2017; Spector & Belongie, 2018; Abel *et al.*, 2018; Shoeleh & Asadpour, 2019). As a result, the researches expressed that *'transfer learning has recently gained popularity due to the development of algorithms that can successfully generalize information across multiple tasks'* (Taylor & Stone, 2011). One of the critical aspects of an intelligent agent is the ability to learn one or multiple environments and transfer previous knowledge to new environments, with similar situations to the previous one. Toward this goal, an autonomous agent must be able to learn first how to behave in a task effectively and then generalize its obtained knowledge as much as needed to transfer and apply in a new domain.

The insight behind TL is that generalization may occur not only within tasks but also across different tasks which are from similar domains. Generally, the idea of the transfer of knowledge is to improve the performance of machine learning algorithms that stem from cognitive science research. A vast number of psychological studies show human beings can learn amazingly fast because they effectively bias the learning process toward a very limited set of solutions obtained by transferring the knowledge retained from solving similar tasks. Similarly, the idea of TL is that it is possible to improve the performance of any machine learning algorithms, like the learning algorithm of autonomous agents, by biasing their hypothesis space toward a set of good hypotheses according to the knowledge retained from solving other tasks.

The main aim of this paper is to facilitate TL for an autonomous agent who can face not only homogeneous problems but also the heterogeneous ones. In general, TL problems can be divided into heterogeneous and homogeneous by considering whether the feature spaces between the source and target domains are the same or not. So, our challenging question is, 'How the source and target tasks are related?'. To answer this question, we propose a novel domain adaptation-based TL approach using an adversarial network, $DATL_{AN}$. It is able to learn a transformation which helps an autonomous RL-based agent to adapt the domains of the source and target task and consequently transfer its skills acquired from source task into the target task.

Domain adaptation is a well-known technique associated with TL which seeks the same goal in machine learning problems, especially pattern recognition. The goal of a domain adaptation approach is to learn and find transformations which can map both source and target domains into a common feature space. On the other hand, generative adversarial networks (*GAN*) (Goodfellow *et al*., 2014) are a promising approach to train a deep network and generate samples across diverse domains. In many applications, these networks can also improve recognition despite the presence of domain changes or data set bias (Liu & Tuzel, 2016; Ganin *et al*., 2016; Tzeng *et al*., 2017). A *GAN* consists of two networks named *generator* and a *discriminator*. They are against each other, means the generator is trained to produce samples with the objective to confuse the discriminator. Recently, one type of domain adaptation approaches which have recently become increasingly popular is known as adversarial adaptation methods. These type of methods seek to minimize an approximate domain discrepancy distance through an adversarial objective with respect to a domain discriminator. They are so closely related to the principles of GAN-based approaches. In domain adaptation, the principle of *GAN* has been employed to ensure that the network cannot distinguish between the distributions of samples coming from the source and target domain (Ganin & Lempitsky, 2014; Liu & Tuzel, 2016; Hoffman *et al*., 2017)

Our proposed method, $DATL_{AN}$, leverages adversarial domain adaptation principles to discover related skills between the source and target tasks, transfer them, and boost the learning performance of the agent in the target task. The $DATL_{AN}$ method has three main steps: first, learning source task and extracting abstract skills by modeling both agent experiences and environment dynamics in *connectivity graph*. Second, finding the state-action inter-task mappings implicitly by leveraging the adversarial domain adaptation technique to learn a common feature space where the source and target domains can be aligned. Then, the agent can efficiently transfer the previously learned skills into the target task in order to learn the new environment. The obtained results from experiments demonstrate that the proposed method is able to find the relation between tasks and consequently transfer effectively skills which were learned in source task. The proposed method improves the performance of an agent in the target task using the transferred knowledge.

The main contributions of this paper can be listed as below:

- The proposed method focuses on the heterogeneous type of transfer problems where the state-action space of the agent is different in source and target tasks.
- The proposed method incorporates domain adaptation techniques into continuous RL domains to automatically learn an inter-task mapping.
- To the best of our knowledge, this paper is the first successful attempt to leverage the GAN for TL in the RL domain.

The rest of this paper is organized as follows. Section 3 presents an overview of the related work. In Section 4, the proposed skill-based TL via domain adaptation approach is described. Experiments and results are reported in Sections 5, and Section 6 contains the conclusion and direction for future works.

2 Background

2.1 Reinforcement learning

RL problems are typically framed in terms of Markov decision processes (MDPs) (Puterman, 2014). For the purposes of this article, MDP and task are used interchangeably. A MDP is defined by four elements named states, actions, transition probabilities, and rewards, as 4-tuple (S, A, P_a, R_a). S is a finite set of states, A is a finite set of actions, $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$, and $R_a(s, s')$ is the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a.

States encode all information of an agent needed to determine how it will evolve when taking actions, with agent governed by the state transition probabilities, $P(s_{t+1}|s_t, a_t)$. The transitions only depend on current state and action, not past states/actions (Markov assumption).

The goal is of RL agent to choose a policy π that will maximize some cumulative function of the random rewards, typically the expected discounted sum over a potentially infinite horizon:

$$E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})\right] \tag{1}$$

where the agent choose $a_t = \pi(s_t)$ (i.e. actions given by the policy), the expectation is taken over $s_{t+1} \sim P_{a_t}(s_t, s_{t+1})$ and γ is the discount factor satisfying $0 \leq \gamma \leq 1$, which is usually close to 1.

So, the solution for an MDP is a policy describing the best action for each state in the MDP, known as the optimal policy π^*. This optimal policy can be found through a variety of methods, like *Q-learning*.

Q-learning is a model-free RL algorithm. It has a function that calculates the quality of a state-action combination, named Q-value ($Q : S \times A \to \mathbb{R}$.). Before agent begins to learn the environment, its Q is initialized to a possibly arbitrary fixed value. Then, at each time t, the agent selects an action a_t, observes a reward r_t, enters a new state s_{t+1} which may depend on both the previous state s_t and the selected action, and the value of Q is updated as below:

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot Q(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \cdot \max_a Q(s_{t+1}, a)\right) \tag{2}$$

where α is the learning rate, γ is the discount factor, s_t is the current state, a_t is the current selected action, r_t is the receiving reward, and s_{t+1} is the next state.

2.2 Transfer learning in reinforcement learning

TL is machine learning with an additional source of information apart from the standard training data: knowledge from one or more related tasks. The goal of TL is to improve learning in the target task by leveraging knowledge from the source task. There are four metrics to find whether transfer improves the learning of an agent or not.

1. *Jumpstart:* the improvement of an agent at the initial performance in a target task.
2. *Asymptotic Performance:* the final performance of a learned agent in a target task.
3. *Transfer Ratio:* the ratio of the total accumulated reward by the agent benefiting TL to the total accumulated reward by the agent without TL.
4. *Time to threshold:* the difference of learning time in terms of episodes needed by the agent to achieve a pre-specified performance level in both source and target tasks.

Each metrics has drawbacks, and none are sufficient to fully describe the benefits of any transfer methods.

Taylor and Stone (2009) and Lazaric (2012) categorize the TL methods in RL domain over three aspects: the setting, the transferred knowledge, and the objective. The authors claimed that the definition of transferred knowledge and the exact transfer process, that is, setting, is the main aspects to characterize a TL algorithm. As the type of transferred knowledge can be primarily characterized by its level of specificity, the possible knowledge transfer approaches can be classified into two main categories accordingly: low-level knowledge transfer and high-level knowledge transfer. In RL domain, low-level information can be considered as (s, a, r, s') tuples, an action-value function Q, a policy π, or a full model of the task, whereas high-level information can be considered as a subset of all actions used in some situations or partial policies, skills or options, rules, essential features for learning, proto-value functions (Mahadevan & Maggioni, 2007), shaping rewards, or subtask definition.

As Taylor and Stone (2009, 2011) claimed, it makes intuitive sense that high-level knowledge may transfer better across tasks since they can be obtained more independently compared to low-level information. Low-level knowledge can all be directly leveraged to initialize a learner in the target task. On the other hand, high-level information may not directly be applicable to TL algorithms to fully define an initial policy for the agent in the target task. However, such information would guide the agent during its learning in the new target environment. Moreover, heterogeneous TL algorithms using high-level knowledge assist the agent in learning a new different task more effectively than lower-level information.

3 Related work

Please note that the proposed method tries to transfer high-level knowledge, namely, a set of skills which were acquired from source task into the target task.

By transferring skills, our method tends to detect and transfer similar region among source and target task. The idea of transferring similar regions among tasks was firstly proposed by Lazaric *et al.* (2008) and Lazaric and Restelli (2011) where the similar regions are determined using the similarity between samples in source and target, indeed using low-level knowledge. In contrast, our proposed method tries to transfer similar regions identified with high-level knowledge. Asadi and Huber (2007, 2015) present an agent that learns options and transfers them between different tasks. The agent tries to find subgoals in the source task through identifying states that are 'locally from a significantly stronger *attractor* for state-space trajectories' (Asadi & Huber, 2007). Considering such subgoals helps the agent define options. The authors assumed that the source and target tasks differ only in the reward function, while the proposed method would be applied to the source and target tasks that may differ in possible state transitions and state-action space.

Da Silva and Costa (2019) categorize the main lines of research on TL for Multiagent RL area. They consider two types of TL: (1) Intra-Agent Transfer, where the agent reuses the knowledge previously generated by itself in new tasks or domains, (2) Inter-Agent Transfer, where the agent tries to find how to best reuse knowledge received from communication with another agent with different sensors and possibly state-action space. According to this categorization, our proposed method belongs to the first category because the agent reuses its own knowledge across different tasks.

As suggested by Lazaric (2012), TL approaches in RL problems can be categorized based on the number of involving source tasks and the difference between source and target domains: (1) transfer from one source task to one target task with fixed domain, (2) transfer across several tasks (including a set of source tasks) with fixed domain, and (3) transfer across several tasks with different domains. The author categorizes TL approaches that the agent reuses its own knowledge. As a principle, it is stated that 'the domain of a task is determined by its state-action space, while the specific structure and goal of the task are defined by the dynamics and rewards'. According to this definition, the first and second categories consist of problems whose state-action spaces are the same. In contrast, source and target tasks in the third category have different domains, meaning different state-action variables. While this category is more common in real-world problems, it involves one additional challenging issue to define the mapping between the source and target state-action variables. In literature, such mappings are referred to as *Inter-task Mappings* which was formulated firstly by Taylor *et al.* (2007). According to the presented categories, our proposed method lies in the third case. It leverages a domain adaptation neural network to find the inter-task mapping between source and target tasks driven from different domains.

Reusing the solution of previous tasks is the most intuitive way to apply TL to RL. In other hand, many methods like the one proposed by Da Silva *et al.* (2017) focused on reusing knowledge from external sources, such as demonstrations from humans. In literature, most of the researchers assume that experts predefined the inter-task mappings according to their experience or intuition. Some researchers design mechanisms to select proper mappings from several predefined mappings. Fachantidis *et al.* (2011) proposed two algorithms to select the best mapping from multiple mappings for both model-based and model-free RL algorithms to transfer from multiple inter-task mappings. Similarly, Fachantidis *et al.* (2015) proposed a method to autonomously select mappings from the set of all possible inter-task mappings. Cheng *et al.* (2019) proposed a many-to-one mapping for TL, named linear multi-variable mapping. It uses the linear combination of the information from different related state variables and action to initialize the target task learning. However, their approach still requires an expert to provide the parameters of the linear combination, and the optimal parameter values are not easy to be given. Da Silva and Costa (2017) estimate an Inter-Task Mapping from a specialized task description using object-oriented MDPs. The author argued that the domain knowledge contained in the task description allows agent to estimate the mapping function without any interaction with the environment in the target task before transfer.

There are also some methods for learning inter-task mapping automatically. Celiberto *et al.* (2011) used a Neural Network to map actions from the source domain to the target domain by observing the results of the two different actions in the source domain and target domain so as to learn the weights of the network. However, the mapping between the states is predefined by an expert. On the other hand, Cheng *et al.* (2017) proposed an artificial neural network-based method to learn both action and state inter-task mapping between source task and target task. The obtained inter-task mapping is used to transfer the knowledge acquired in the source task into the target task for initialization.

The closest approaches related to our work are the approaches proposed by Ferns *et al.* (2011) and Ammar *et al.* (2014). They were trying to find an inter-task mapping for a pair of tasks or finding the MDP similarities to have effective TL approach. Ammar *et al.* (2012) proposed a TL framework which learns the inter-task mapping by representing the source and target data, in the form of (s, a, s'), in a high-dimensional space discovered using sparse coding, projection, and Gaussian process.

Ammar *et al.* (2012, 2015) proposed a TL method in the context of policy gradient RL. The multi-task learning method proposed by Ammar *et al.* (2015) transfers the shared knowledge between sequential decision-making tasks by incorporating latent basis into policy gradient learning. Ammar *et al.* (2012) proposed a system to transfer the source samples into the target by discovering a high-level feature space through learning inter-task mapping via an unsupervised manifold alignment. Similarly, Bocsi *et al.* (2013) proposed an alignment-based TL method for robot models. The primary differences with our work are that they focused on transferring models or policies/samples between different tasks, rather than high-level knowledge, that is, skills, therefore the authors needed a similarity metric for MDPs. Despite the invaluable research done for TL in the RL realm, to the best of our knowledge, there are still open directions in this area to transfer autonomously learning without requiring any background knowledge.

4 Domain adaptation-based transfer learning using an adversarial network

In our proposed method, the autonomous agent uses a domain adaptation technique to discover a mapping that can align the state-action spaces of the new environment to the one which was learned previously. This mapping is called inter-task mapping between state-action spaces of the source and target environments. Here, we utilize the concept of domain adaptation technique to facilitate TL across domains with different state-action spaces. Our proposed agent must perform three learning phases: (1) learning source task, (2) learning similarities between source and target tasks, and (3) learning target task.

4.1 Learning source task

As the first step, the agent should learn the source task properly and its experiences must be captured as high-level knowledge such as skills in order to be appropriately transferred. Many approaches have been proposed to extract skills in RL realm. Among them, we suggest using Graph-based Skill Learning method (*GSL*), which is proposed by Shoeleh and Asadpour (2017). The promising results demonstrated that *GSL* approach not only can find appropriate skills but also its results show notable improvements in the learning performance of the agent. The agent's experiences are captured as a *connectivity graph*, which gives information about both the agent's dynamic behavior and the environment's dynamics. The communities found from such graph divide the state-space into regions called *accessible regions*, and the agent learns the problem by extracting a skill for each *accessible regions*. *GSL* accomplishes hierarchical learning by decomposing a problem into the set of skills and then benefits the *option framework* which were proposed by Sutton *et al*. (1999) to learn those skills.

4.2 Learning similarities between source and target tasks

After learning the source task, the next step is determining 'How the two tasks with different state variables and actions are related?' A possible way to answer this question is finding a common latent space where the source and target state-action spaces can be aligned. Domain adaptation is a well-known technique which seeks the same goal in pattern recognition. Considering a classification task where X is the input space and $Y = \{0, 1, ..., L-1\}$ is the set of L possible labels. Moreover, there are two different distributions over $X \times Y$, called the *source domain* D_S and the *target domain* D_T. An *unsupervised domain adaptation* learning algorithm is then provided with a *labeled source sample* S drawn *i.i.d.* from D_S, and an unlabeled target sample T drawn *i.i.d.* from D_T^X, which is the marginal distribution of D_T over X. $S = \{(x_i; y_i)\}_{i=1}^n \sim (D_S)^n$; $T = \{(x_i)\}_{i=n+1}^N \sim (D_T^X)^{n'}$ with $N = n + n'$ being the total number of samples. The goal is to build a classifier $\nu : X \to Y$ with a low *target risk* while having no information about the labels of D_T.

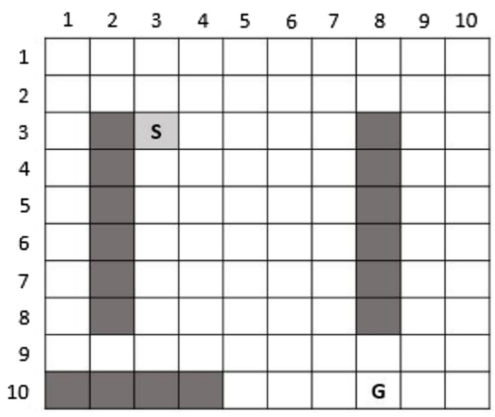
In the following, we detail how to develop and feed a GAN to find the inter-task mapping, align the samples of the source and target tasks to each other, and consequently transfer the skills which were learned before into the new environment. To do so, we offer to adapt the state-of-the-art approach called domain-adversarial neural network (*DANN*) proposed by Ganin *et al*. (2016). This technique incorporates a domain adaptation component to neural networks. We feed *DANN* with two following sets of samples, S and T which are collected from the source and target domain, respectively:

$$\begin{aligned} S = \{(x_i^S, y_i^S)\}_{i=1}^n \qquad & \textit{where} \\ & x_i^S = < s^S, a_1^S, s'_{a_1^S}, r_{a_1^S}, Q^{sS}_{a_1^S}, a_2^S, s'_{a_2^S}, r_{a_2^S}, Q^{sS}_{a_2^S}, ..., a_k^S, s'_{a_k^S}, r_{a_k^S}, Q^{sS}_{a_k^S} > \\ & y_i^S = ID \text{ of the skill that sample(state) } s^S \text{ located in.} \end{aligned} \tag{3}$$

$$\begin{aligned} T = \{(x_i^T, y_i^T)\}_{i=n+1}^N \qquad & \textit{where} \\ & x_i^T = < s^T, a_1^T, s'_{a_1^T}, r_{a_1^T}, Q^{sT}_{a_1^T}, a_2^T, s'_{a_2^T}, r_{a_2^T}, Q^{sT}_{a_2^T}, ..., a_k^T, s'_{a_k^T}, r_{a_k^T}, Q^{sT}_{a_k^T} > \end{aligned} \tag{4}$$

where x_i^S is the ith sample collected from the source domain. This sample represents the current state s^S, the state transition (doing action a_j^S in the source environment makes the state of agent change into $s'_{a_j^S}$, where $j \in [1, 2, .., k]$ and k is the possible number of actions can be chosen in the source environment), the given reward from the source environment by choosing jth action $r_{a_j^S}$, and the q-value $Q^{sS}_{a_j^S}$ which indicates the value of choosing a_j^S in the current state, s^S. y_i^S indicates the ID of the skill in which the current state s^S is located. Similarly, x_i^T represents the ith instance sampled from the target domain. Figure 1 illustrates an example of sample representation in Maze environment. Please note that in our setting, for each sample in source domain y_i^S is discovered through learning the source task, but y_i^T is not defined yet in the target domain. Therefore, we utilize *DANN* as an unsupervised domain adaptation to determine the y_i^T in the target domain by adapting the source and target domains. Its architecture is shown in Figure 2. It includes three deep neural networks: a deep feature extractor, a deep skill predictor, and a domain classifier.

DANN is motivated and supported by the theory on domain adaptation presented by Ben-David *et al*. (2007, 2010), '*a good representation for cross-domain transfer is one for which an algorithm cannot learn to identify the domain of origin of the input observation*'. So, the unsupervised domain adaptation architecture (Figure 2) focuses on extracting and learning a feature set which combines both



X = [(3,3),
up, (3,2), −1 , 87,
right, (4,3), −1, 89,
down, (3,4), −1 ,89 ,
left, (3,3), −10, 88]

Figure 1. Example of sample representation in Maze environment

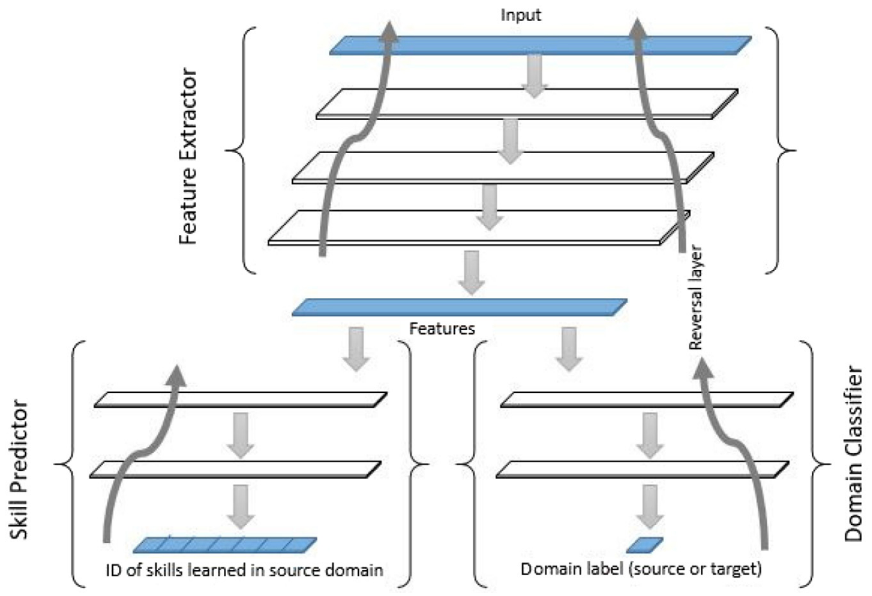


Figure 2. Unsupervised domain adaptation architecture includes a deep feature extractor, a deep skill predictor, and a domain classifier connected to the feature extractor via a gradient reversal layer. Gradient reversal results in the domain-invariant features by ensuring that the feature distributions over the two domains are made similar and simultaneously as indistinguishable as possible for the domain classifier

discriminativeness (discriminative for learning the skill of a given state in source environment) and domain-invariance (invariant to the change of domains). It jointly optimizes the underlying feature set as well as two discriminative classifiers operating on this feature set: (1) *Skill predictor* predicting *ID* of the skill that a given state located in and (2) *Domain classifier* discriminating between the source and the target domains. As proposed by Ganin *et al*. (2016), *DANN* uses standard layers and loss functions. It trains using standard backpropagation algorithms based on stochastic gradient descent or its modifications (e.g. *SGD* with momentum). The domain classifier is connected to the obtained feature set via a gradient reversal layer resulting in the domain-invariant features, means the feature set distribution over the two domains are made similar and as indistinguishable as possible to classify the domain. The optimization of training *DANN* is as follows:

$$E(\theta_f, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_y^i(\theta_f, \theta_y) \\ - \lambda \left(\frac{1}{n} \sum\nolimits_{i=1}^{n} \mathcal{L}_d^i(\theta_f, \theta_y) + \frac{1}{n'} \sum\nolimits_{i=n+1}^{N} \mathcal{L}_d^i(\theta_f, \theta_y) \right) \quad (5)$$

As suggested by Ganin *et al.* (2016), the saddle point optimizing above equation can be found as a stationary point of the following gradient updates:

$$\theta_f \leftarrow \theta_f - \mu \left(\frac{\partial \mathcal{L}_y^i}{\partial \theta_f} - \lambda \frac{\partial \mathcal{L}_d^i}{\partial \theta_f} \right) \tag{6}$$

$$\theta_y \leftarrow \theta_y - \mu \left(\frac{\partial \mathcal{L}_y^i}{\partial \theta_y} \right) \tag{7}$$

$$\theta_d \leftarrow \theta_d - \mu\lambda \left(\frac{\partial \mathcal{L}_d^i}{\partial \theta_d} \right) \tag{8}$$

where μ is the learning rate. Stochastic estimates of these gradients are used by sampling examples from the data set.

4.3 Learning target task

In our proposed approach, the output of first learning phase is a set of skills extracted and learned from the source task and the output of second learning phase is a learned deep neural network which identifies the state-action space mappings. Since in RL literature, skills can be formulated using *Option framework* as a well-defined temporal abstraction frameworks extending RL algorithms from primitive actions to time extended activities, our agent uses this framework not only to learn skills in source task and construct its own high-level skill hierarchy but also to transfer them into target task. So, for each skill, an option O is created and its properties, namely the termination condition O_T, reward function O_R, initiation set O_I, and its internal function approximator, should be defined or learned. Following skill *id* assignment to each sample of the target using GANN, the termination condition O_T, initiation set O_I, and reward function O_R of each option can be defined as suggested by Shoeleh and Asadpour (2017). However, the function approximator of a skill cannot be directly transferred because its parameters were learned in the source domain whose state-action spaces are different from the target's ones. To successfully transfer the function approximators, the agent learns skills' function approximators offline by utilizing the output of *GANN* and the *q-value* of samples in source task (see Algorithm 1). To negative transferring, we suggest a *KNN*-based mechanism to eliminate samples whose k nearest neighbors do not have the same skill *id*. These samples are considered as negative samples; indeed, they may be noisy or border samples, and in both cases, it is better to eliminate them.

In spite of the previous researches that have initiated new option policies using the past experiences, these further updates may be experimentally confounding. Therefore, we would not directly add the transferred skills to the agent's action repertoire. These skills are firstly considered as gestating skills. This type of skills have a gestating period time (e.g. 10 episodes). During this period, they cannot be selected for execution, but their policies are updated using off-policy learning. Each gestating skill finishing its gestating period would be added to the agent's action set as a learned skill and assign appropriate initial values as its value. The initial value of a new transferred skill is considered as the maximum of Q values of its border states estimated during the gestating period.

In addition, our proposed method tends to improve the performance agent by transferring the previously learned skills into the new domain. So, it is necessary to find which learned skills, mapped from source to target task through domain adaptation, are admissible for transfer. To answer this question, we calculate a matching-based fitness for all mapped and learned skills in the target task. The eligibility of each skill is defined as a ratio of its region size in the target task to its region size in source task. If the fitness of one skill is greater than a threshold θ, it is considered as admissible for transferring, and agent expands its action-value function to include this skill.

5 Experimental results

We evaluate the performance of our proposed method, namely $DATL_{AN}$, through several experiments. In the following, we first introduce the test domain and then present the experiments and evaluations on our approach for TL.

Algorithm 1 Transfer Learning

```
1:  $D_S$: $\{\tau_s = < s_s, a_s, s'_s, r_s >\}$ collected form the source task.
2:  $D_T$: $\{\}\tau_t =< s_t, a_t, s'_t, r_t >\}$ collected form the target task.
3:  $Q_S$ : q-value of the source samples $< (s_s, a_s), q >$
4:  $\tau_s = M(\tau_t)$: similarity mapping function using GANN to find a sample($\tau_s$) of the source task, which
    is most similar to the input target task sample($\tau_t$)
5:  $id = Skill_{ID}(\tau_t)$ : The skill $id$ assignment function returns the $id$ of the skills in where the target
    sample $\tau_t$ is located
6:  $O$: a list raw skills defined in target task using skill $id$ assignment function. Each $o \in O$ has a
    function approximator with $\theta_{o,FA[a]}$ parameters

7:  **For** each sample $\tau_t \in D_T$
8:    $\tau_s = M(\tau_t)$
9:    $id = Skill_{ID}(\tau_t)$
10:   $o = O.get(id)$
11:   $R = Q_S(s_s, a_s)$ where $(s_s, a_s) \in \tau_s$
12:   $Q(s_t, a_t) = \theta^T_{o.FA[a_t]}\phi[s_t]$
13:   $(\theta_{o.FA[a_t]}, z_{a_t}) = \text{TD}(R, Q_{s_t,a_t}, 0, \theta_{o.FA[a_t]}, z_{a_t})$

14: **Repeat** (for each episode $e$) until converge
15:   Initialize all $z_o = 0$ for each $o \in O$
16:   $E = \{(e', o')\}$ :splitting $e$ in such that $e'$ is a sub-episode where the option $o'$ is active
17:   **For** each $(e', o') \in E$
18:     **For** each step $(s, a, r, s', a')$ of $e'$:
19:       $Q_{s,a} = \theta^T_{o'.FA[a]}\phi[net_S(s)]$
20:       $Q_{s',a'} = \theta^T_{o'.FA[a']}\phi[net_S(s')]$
21:       $(\theta_{o'.FA[a]}, z_a) = \text{TD}(R, Q_{s,a}, Q_{s',a'}, \theta_{o'.FA[a]}, z_a)$

22: **function** TD $(r, Q_{(s,a)}, Q_{(s',a')}, \theta_{FA[a]}, z)$
        $\delta = r + \gamma * Q_{(s',a')} - Q_{(s,a)}$
        $z = \theta_{FA[a]} + \gamma.\lambda.z$
        $\theta_{FA[a]} = \theta_{FA[a]} + \alpha.\delta.z$
23: **end function**
```

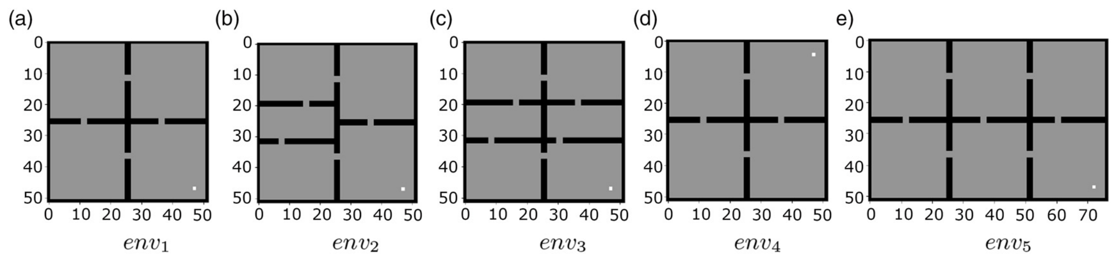


Figure 3. *Four Room* environments with different locations of obstacles and environment's state space

We examine our proposed approach using two well-known benchmarks in RL, named *Four Room* as illustrated in Figure 3, and a real-time strategy game named *StarCraft:Broodwar*.

Four Room: The problem space is four neighbor rooms which are connected with four doors. The agent's discrete state space is shown with grids. In each state, four primitive actions are available: moving up or down, turning left or right. If doing each action leads to a wall hit, there is no change in the agent's state and it stays in its previous state. Each episode starts from a start state which is chosen randomly in the start of each episode and finishes when the agent reaches a goal state, which is fixed in all episodes and is shown in green in Figure 3. The agent receives a reward -1 for performing each action, -10 for hitting the wall, and $+100$ for reaching the goal state. Here, to examine our proposed approach, we use a set of different *Four Room* problems which are different in the locations of obstacles and environment's state space as illustrated in Figure 3.

StarCraft:Broodwar: In this game, the agent should fight with several enemies spreading around the environment. The game ends either when the agent kills all enemies or be killed. Both RL agent and enemies have hit points which shows their healthiness. So, if one unit reaches zero hit point, it means that it is killed and should be eliminated. The state-space of the problems depends on the number of enemies. It consists of the locations of each enemy and the location of the RL agent, plus the hit points and the weapon ranges. The agent can perform three possible actions: (1) moving toward a target enemy: if the agent collides with obstacles, its location does not change. (2) retreating: the RL agent moves one step toward its initial position where it starts the game. (3) firing at a target enemy: if the enemy is within the weapon range of the agent, the agent will fire at it, and the bullet would hit the enemy with the probability 0.7. If the agent can successfully shoot at an enemy, the hit points of the enemy would decrease by 1. Otherwise, its hit points do not change. This action does not change the location of the agent.

The reward which the agent would receive by doing action a from state s and going to state s' is computed as follows:

$$r = 10 \times \left(\sum_{i=1}^{\#enemies} (HP_i(s) - HP_i(s') - (HP(s) - HP(s')) + ra - 1) \right) \tag{9}$$

where HP and HP_i show the hit point of the agent and the ith enemies, respectively, and $ra \sim N(-10, 0, 1)$.

In order to evaluate the effectiveness of our approach, we compare three learners: (1) a standard agent applying $SARSA(\lambda)$ with linear function approximation using Fourier basis (Konidaris *et al.*, 2011) as a standard RL method, (2) an agent with the ability of option learning introduced by Shoeleh and Asadpour (2017), named *GSL* agent without using TL, and (3) an agent using the proposed method, named $DATL_{AN}$. These agents are configured as 1000 learning episodes, learner Fourier order and option Fourier order are 5 and 3, respectively, λ is 0.9, and α decreases adaptively (Dabney and Barto, 2012). It is worth mentioning that like the standard agent, both *GSL* and $DATL_{AN}$ agents use linear function approximation with Fourier basis functions. Since each option covers a subspace of the whole problem space, the Fourier order of option's function approximator is smaller than the agent's function approximator. Note that the first 10 episodes of the learning phase of *GSL* and $DATL_{AN}$ agents are devoted to gathering experiences with random policy (ϵ-greedy with $\epsilon = 1$) to construct the *connectivity graph* and collect a set of samples used in domain adaptation, respectively. Note that we use the released source code for the Gradient Reversal layer as an extension to Caffe (Ganin & Lempitsky, 2015).[1]

To examine and appraise our approach $DATL_{AN}$, we consider four TL scenarios using *four room* benchmark to transfer the learned skills from an environment as source task to a new environment as a target task. Similarly, we consider three TL scenarios using *start craft* benchmark:

- S_1 : transferring from env_1 to env_2.
- S_2: transferring from env_1 to env_3.
- S_3: transferring from env_1 to env_4.
- S_4: transferring from env_1 to env_5.
- S_5: transferring from $game_1$ where the position of enemy is [3, 10] to $game_2$ where the position of the enemy changes to [10, 10].

[1] https://github.com/ddtm/caffe/tree/grl

Table 1. Comparison of $DATL_{AN}$, *GSL*, and *SARSA* agents in Four-Room benchmarks in terms of obtained *Return*, average number of *Steps* to reach goal state and average number of *Interaction* between agent and the environment

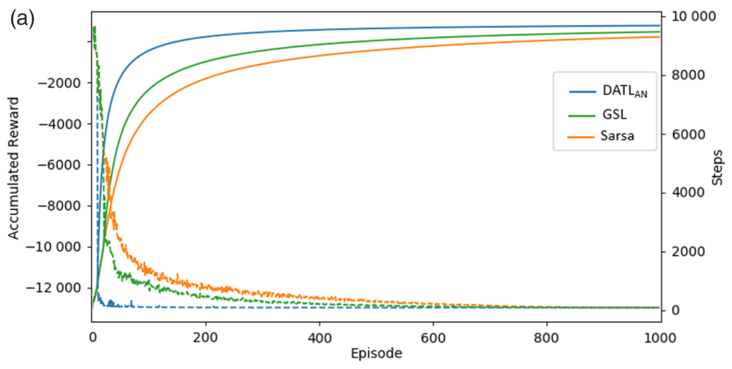
		$DATL_{AN}$	*GSL*	*SARSA*
S_1	Return	916.72	916.6	916.11
	# Steps	83.28	83.4	83.89
	# Interactions	**141 095**	448 255	653 243
S_2	Return	906.8	906.41	906.3
	# Steps	93.2	93.59	93.7
	# Interactions	**206 715**	509 037	758 677
S_3	Return	916.7	916.64	916.15
	# Steps	83.3	83.36	83.85
	# Interactions	**210 309**	449 078	653 615
S_4	Return	890.3	862.4	836.9
	# Steps	109.7	137.6	163.1
	# Interactions	**482 422**	761 848	857 893

- S_6 : transferring from $game_3$ where the position of enemies are [4, 8] and [10, 10] to $game_4$ where the position of the enemy changes to [3, 10] and [10, 10].
- S_7 : transferring from $game_5$ where the position of enemies are [4, 8], [10, 10], and [10, 3] to $game_6$ where the position of the enemy changes to [3, 10], [10, 3], and [10, 10].

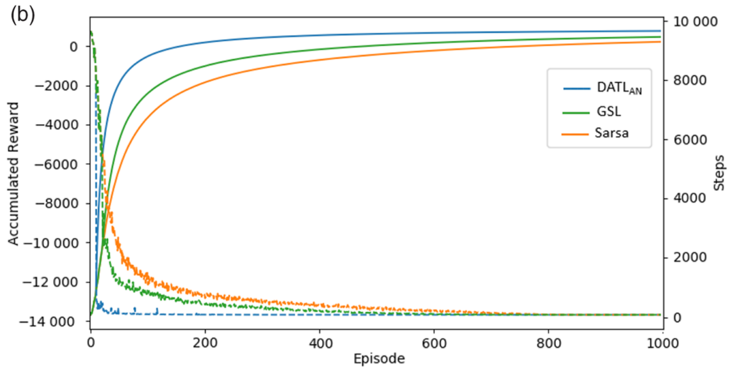
Please note that in all start craft games, the position of RL agent is [3, 3], the total hit point of the agent is 10 and each enemy has 3 hit points.

Figure 4 illustrates the performance of three agents considering four scenarios with a different configuration of *four Room* problem as-mentioned above. The obtained results highlight the competitiveness of our methods in terms of the obtained accumulated rewards during the time to other learners. Besides, Table 1 shows the comparison of these agents in terms of obtained *Return*, the average number of steps to reach the goal state and the average number of interaction needed between agent and environments to learn the target task. Decreasing the steps needed to reach the goal state means decreasing interactions of the agent with the environment and increasing learning speed. As expected, the agents, which uses skills, (*GSL* and $DATL_{AN}$) learns optimal policies with fewer experiences than the standard agent. In addition, $DATL_{AN}$ agent needs less interaction with the environment since it benefits TL. For example, *GSL* and $DATL_{AN}$ agents can, respectively, learn env_2 in the first scenario with nearly 69% and 22% of the number of interactions which *SARSA* agents needs. Similarly, in the fourth scenario, agents *GSL* and $DATL_{AN}$ need nearly 89% and 56% the number of interactions *SARSA* agents needs to learn env_5, respectively. Since in the fourth scenario, the agent faces heterogeneous TL problem, the ratio of the agent's interactions decreasing is less than to the one in the first scenario where there is a homogeneous TL problem. The results presented in Figure 4 and Table 1 indicate that $DATL_{AN}$ agent outperforms *GSL* one, the agent without utilizing a TL technique, in terms of mentioned metrics.

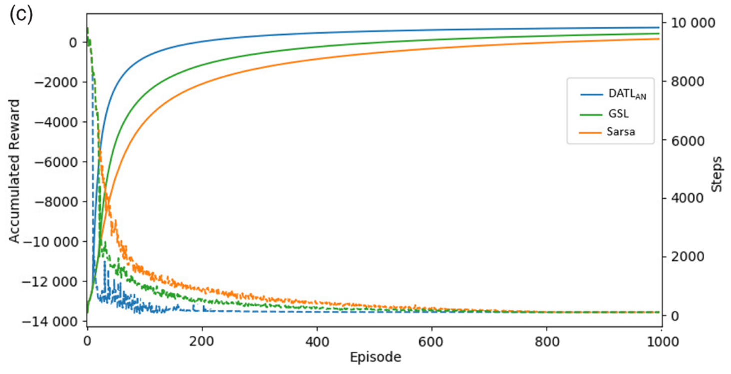
To figure out the effectiveness of the proposed method in continuous RL domain, three scenarios which are designed using an abstract combat maneuvering model of StarCraft:Broodwar game. The performance of $DATL_{AN}$ agent is compared with three other agents named *GSL*, *SARSA*, and *SCL* as a state-of-the-art method in skill learning in continuous RL domain (Konidaris *et al.* 2012). The obtained results considering three scenarios are presented in Table 2 and Figure 5. The results illustrated that whenever the problem becomes harder, the number of interactions needed for the RL agent to learn the policy in the environment would be more. For example, scenario $S7$ is the hardest problem because the number of enemies is more and consequently, the problem space would be higher. The results presented in Table 2 and Figure 5 indicate that TL helps the agent to learn the problem with less number of interactions in comparison to not using TL. Also, $DATL_{AN}$ agent can outperform other agents in addition to



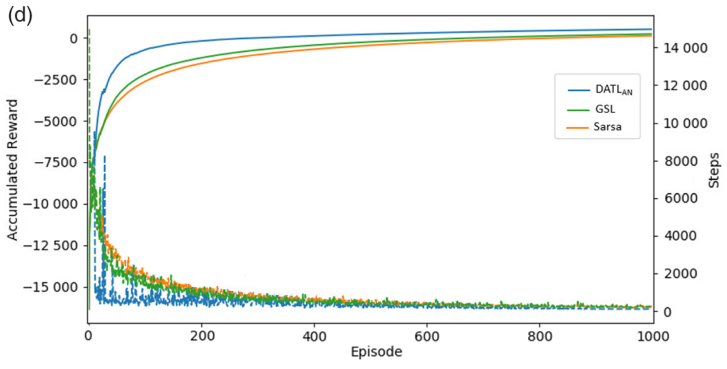
S_1: Transferring from env_1 to env_2



S_2: Transferring from env_1 to env_3

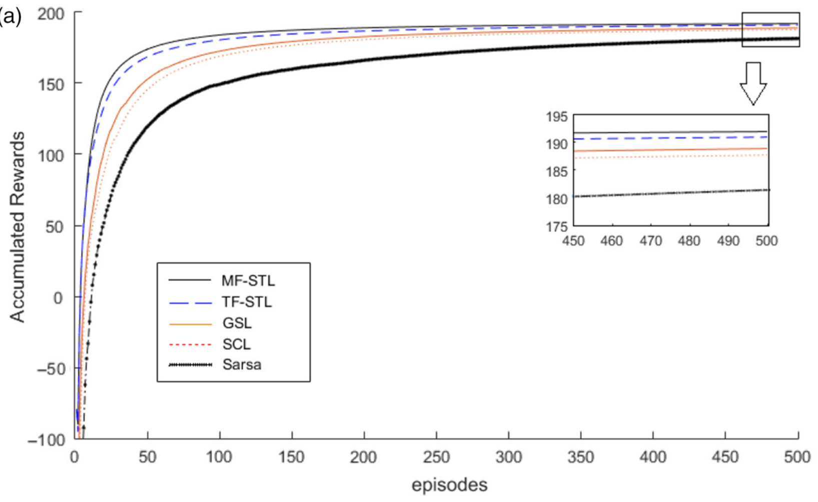


S_3: Transferring from env_1 to env_4

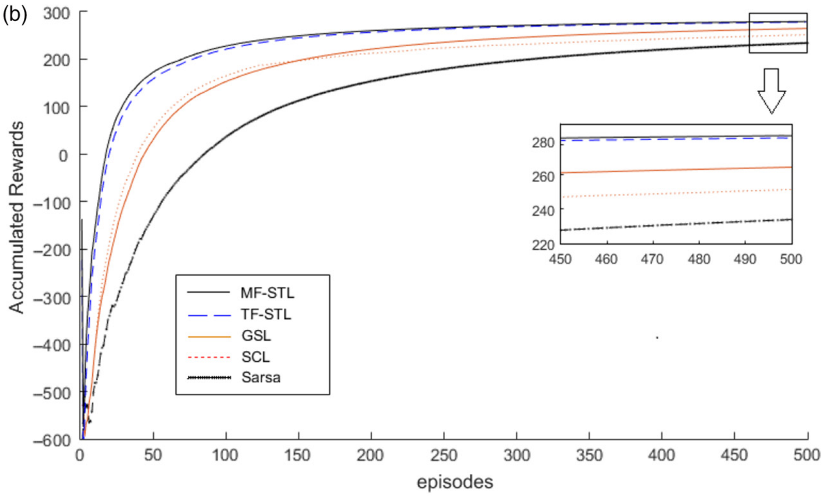


S_4: Transferring from env_1 to env_5

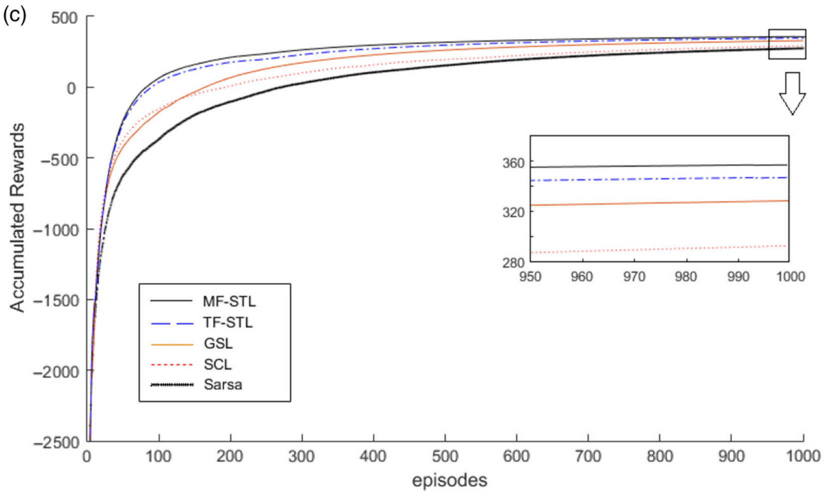
Figure 4. Comparison of learning the performance of three agents *Sarsa*(orange), *GSL* (green), and $DATL_{AN}$ (blue) in terms of accumulated reward (solid line) and the number steps (dashed line) for the four scenarios defined in the four-room problem

(a)


S_5: Transferring from $game_1$ to $game_2$

(b)


S_6: Transferring from $game_3$ to $game_4$

(c)


S_7: Transferring from $game_5$ to $game_6$

Figure 5. Comparison of learning performance of five agents: *Sarsa*, *GSL* (Shoeleh & Asadpour, 2017), *SCL* (Konidaris *et al*., 2012), and two agents using proposed method with matching-based fitness ($MF - DATL_{AN}$) or trajectory-based fitness ($TF - DATL_{AN}$) in the three scenarios defined in the StarCraft:Broodwar problem.

Table 2. Comparison of *DATL*$_{AN}$, *GSL*, *SCL*, and *SARSA* agents in StartCraft game as a continuous RL problem in terms of obtained *Return*, average number of *Steps* to reach goal state, and average number of *Interaction* between agent and the environment

		DATL$_{AN}$	*GSL*	*SCL*	*SARSA*
S_5	Return	199(±0.89)	199(±1.05)	199(±2.31)	197(±4.07)
	# Steps	11(±0.96)	11(±1.06)	11(±0.23)	13(±1.63)
	# Interactions	**1719**(±314)	3822(±518)	3958(±892)	5046(±1161)
S_6	Return	322(±1.76)	320(±3.86)	319(±5.08)	317(±7.15)
	# Steps	18(±1.04)	20(±1.76)	21(±4.01)	23(±9.11)
	# Interactions	**6569**(±481)	11 248(±2491)	12 622(±2703)	14 572(±2975)
S_7	Return	442(±5.91)	430(±10.4)	432(±18.76)	423(±24.91)
	# Steps	28(±1.21)	31(±3.11)	36(±7.19)	47(±13.89)
	# Interactions	**13 512**(±1078)	52 466(±7520)	65 059(±8178)	79 168(±10621)

Table 3. Average and standard deviation of metrics (introduced by Taylor and Stone (2009)) for proposed method over 30 independent runs

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
Jumpstart	1286 (±208)	**2976** (±231)	679 (±357)	1264 (±392)	118 (±41)	567 (±37)	183 (±49)
Asymptotic performance	**817.55** (±11)	697.38 (±15)	808.39 (±19)	483.37 (±21)	190.4 (±10)	281.8 (±19)	348.1 (±29)
Transfer ratio	13.98 % (±0.01)	**16.95** % (±0.02)	13.86 % (±0.02)	15.37 % (±0.01)	7.06 (±0.02)	12.01 (±0.01)	9.46 (±0.02)
Time to threshold	232 (±19)	441 (±31)	528 (±56)	**631** (±71)	238 (±31)	231 (±34)	447 (±29)

learning the problem with a better amount of decrease in the needed number of interactions, especially when facing a harder problem.

In this paper, we use four metrics introduced by Taylor and Stone (2009) to measure the benefits of transfer. Although these metrics seem implicitly evident in Figure 4 and Table 1, they are explicitly outlined in Table 3. Note that in calculation of *Time to threshold* metric, the *asymptotic performance* of *GSL* agent is defined as threshold. According to *Jumpstart metric*, using TL makes *DATL*$_{AN}$ agent reach *GSL* agent's performance before 500 and 640 episodes, respectively, in homogeneous and heterogeneous problems, while *GSL* agent achieves this after 1000 episodes. The results indicate that *DATL*$_{AN}$ agent outperforms *GSL* one, an agent without utilizing TL technique, in terms of these metrics. The results obtained in Table 3 demonstrates that the *DATL*$_{AN}$ agent benefits TL to learn target task better in terms of mentioned metrics.

6 Conclusion

In this paper, we proposed a novel domain adaptation-based TL method using adversarial networks, named *DATL*$_{AN}$. A *DATL*$_{AN}$-based agent learns source task by extracting learned skills as high-level knowledge to be leveraged in the new target task. Therefore, we utilize *GSL* framework proposed by Shoeleh and Asadpour (2017), to discover abstract skills as high-level knowledge by constructing *connectivity graph* as a model to capture the agent's experiences and the environment's dynamics. After learning the source task, *DATL*$_{AN}$-based agent deploys well-known domain-adversarial learning named *DANN*, to find a feature space. In this subspace, the transition samples collected from both source and target tasks are related. Consequently, the agent can find which target sample would be located in which learned skill and then assign it the right skill *id*. Having these mappings helps the agent to be able to apply skills that are learned previously in source task into a new but related heterogeneous task. We examined our method in four scenarios containing a well-known four-room test domain in RL. The defined scenarios include either homogeneous or heterogeneous problems. The promising results indicate that transferring skills as a high-level knowledge from the source task to target task by using domain adaptation technique is lucrative. Our current TL approach only considers one source task. It can be extended to utilize high-level knowledge from multiple source domains. One potential solution is to assign different weights to the learned skills obtained from various tasks based on their fitness.

References

Abel, D., *et al.* 2018. Policy and value transfer in lifelong reinforcement learning. In *International Conference on Machine Learning*, 20–29.

Ammar, H. B., Eaton, E., Ruvolo, P. & Taylor, M. 2014. Online multi-task learning for policy gradient methods. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 1206–1214.

Ammar, H. B., Eaton, E., Ruvolo, P. & Taylor, M. E. 2015. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In *Proceedings of AAAI.*

Ammar, H. B., Eaton, E., Taylor, M. E., *et al.* 2014. An automated measure of MDP similarity for transfer in reinforcement learning. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Ammar, H. B., *et al.* 2012. Reinforcement learning transfer via sparse coding. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems,*. **1**. International Foundation for Autonomous Agents and Multiagent Systems, 383–390.

Asadi, M. & Huber, M. 2007. Effective control knowledge transfer through learning skill and representation hierarchies. In *20th International Joint Conference on Artificial Intelligence*, ICML, 2054–2059.

Asadi, M. & Huber, M. 2015. A dynamic hierarchical task transfer in multiple robot explorations. In *Proceedings on the International Conference on Artificial Intelligence (ICAI),* **8**, 22–27.

Barreto, A., *et al.* 2017. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*, 4055–4065.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., *et al.* 2010. A theory of learning from different domains. *Machine Learning* **79**(1–2), 151–175.

Ben-David, S., Blitzer, J., Crammer, K. & Pereira, F. 2007. Analysis of representations for domain adaptation. In: *Advances in Neural Information Processing Systems*, 137–144.

Bocsi, B., Csató, L. & Peters, J. 2013. Alignment-based transfer learning for robot models. In *The 2013 International Joint Conference on Neural Networks (IJCNN),* 1–7. IEEE.

Celiberto Jr., L. A., *et al.* 2011. Using cases as heuristics in reinforcement learning: a transfer learning application. In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence* **22**(1), 1211.

Cheng, Q., Wang, X. & Shen, L. 2017. An autonomous inter-task mapping learning method via artificial neural network for transfer learning. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 768–773. IEEE.

Cheng, Q., Wang, X. & Shen, L. 2019. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research* **64**, 645–703.

Da Silva, F. L. & Reali Costa, A. H. 2017. Towards zero-shot autonomous inter-task mapping through object-oriented task description. In: *Workshop on Transfer in Reinforcement Learning (TiRL).*

Da Silva, F. L. & Reali Costa, A. H. 2019. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research* **64**, 645–703.

Da Silva, F. L., Glatt, R. & Reali Costa, A. H. 2017. Simultaneously learning and advising in multiagent reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 1100–1108. International Foundation for Autonomous Agents and Multiagent Systems.

Dabney, W. & Barto, A. G. 2012. Adaptive step-size for online temporal difference learning. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Fachantidis, A., *et al.* 2011. Transfer learning via multiple inter-task mappings. In European Workshop on Reinforcement Learning, 225–236. Springer.

Fachantidis, A., *et al.* 2015. Transfer learning with probabilistic mapping selection. *Adaptive Behavior* **23**(1), 3–19.

Ferns, N., Panangaden, P. & Precup, D. 2011. Bisimulation metrics for continuous Markov decision processes. *SIAM Journal on Computing* **40**(6), 1662–1714.

Ganin, Y. & Lempitsky, V. 2014. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*.

Ganin, Y. & Lempitsky, V. S. 2015. *Unsupervised domain adaptation by back-propagation*. In ICML.

Ganin, Y., Ustinova, E., *et al.* 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research* **17**(1), 2096–2030.

Goodfellow, I., *et al.* 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2672–2680.

Hoffman, J., *et al.* 2017. Simultaneous deep transfer across domains and tasks. In *Domain Adaptation in Computer Vision Applications*, 173–187. Springer.

Konidaris, G. & Barto, A. G. 2009. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, 1015–1023.

Konidaris, G., Thomas, P., *et al.* 2011. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, 380–385.

Konidaris, G., *et al.* 2012. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research* **31**(3), 360–375.

Lazaric, A. 2012. Transfer in reinforcement learning: a framework and a survey. *Reinforcement Learning* **12**, 143–173.

Lazaric, A. & Restelli, M. 2011. Transfer from multiple MDPs. In *Advances in Neural Information Processing Systems*, 1746–1754.

Lazaric, A., Restelli, M. & Bonarini, A. 2008. Transfer of samples in batch reinforcement learning. In: *Proceedings of the 25th International Conference on Machine Learning – ICML 2008,* pp. 544–551. ACM Press.

Liu, M.-Y. & Tuzel, O. 2016. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems*, 469–477.

Mahadevan, S. & Maggioni, M. 2007. Proto-value functions: a Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research* **8**, 2169–2231, 16.

Moradi, P., *et al.* 2012. Automatic skill acquisition in reinforcement learning using graph centrality measures. *Intelligent Data Analysis* **16**, 113–135.

Puterman, M. L. 2014. *Markov Decision Processes*: *Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Shoeleh, F. & Asadpour, M. 2017. Graph based skill acquisition and transfer Learning for continuous reinforcement learning domains. *Pattern Recognition Letters* **87**, 104–116.

Shoeleh, F. & Asadpour, M. 2019. Skill based transfer learning with domain adaptation for continuous reinforcement learning domains. *Applied Intelligence*, 1–17.

Spector, B. & Belongie, S. 2018. Sample-effcient reinforcement learning through transfer and architectural priors. *arXiv preprint arXiv:1801.02268*.

Sutton, R. S. S., Precup, D. & Singh, S. 1999. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* **112**(1–2), 181–211.

Taylor, M. E. & Stone, P. 2009. Transfer learning for reinforcement learning domains: a survey. *Journal of Machine Learning Research* **10**, 1633–1685.

Taylor, M. E. & Stone, P. 2011. An introduction to intertask transfer for reinforcement learning. *AI Magazine* **32**(1), 15.

Taylor, M. E., Stone, P. & Liu, Y. 2007. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research* **8**, 2125–2167.

Tzeng, E., *et al.* 2017. Adversarial discriminative domain adaptation. *Computer Vision and Pattern Recognition (CVPR)* **1**(2), 4.