IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 September 2024, accepted 1 November 2024, date of publication 13 November 2024,
date of current version 5 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3497589

RESEARCH ARTICLE

Knowledge Transfer in Deep Reinforcement Learning via an RL-Specific GAN-Based Correspondence Function

MARKO RUMAN[1] AND TATIANA V. GUY[1,2], (Senior Member, IEEE)
[1]Department of Adaptive Systems, Institute of Information Theory and Automation, Czech Academy of Sciences, 182 00 Prague, Czech Republic
[2]Department of Information Engineering, Faculty of Economics and Management, Czech University of Life Sciences, 165 00 Prague, Czech Republic

Corresponding author: Marko Ruman (ruman@utia.cas.cz)

This work was supported in part by the Joint Ústav teorie informace a automatizace (UTIA)-Provozně ekonomická fakulta (PEFT) Laboratory TALISMAN, and in part by the European Cooperation in Science and Technology through COST Action under Grant CA21169.

ABSTRACT Deep reinforcement learning has demonstrated superhuman performance in complex decision-making tasks, but it struggles with generalization and knowledge reuse—key aspects of true intelligence. This article introduces a novel approach that modifies Cycle Generative Adversarial Networks specifically for reinforcement learning, enabling effective one-to-one knowledge transfer between two tasks. Our method enhances the loss function with two new components: model loss, which captures dynamic relationships between source and target tasks, and Q-loss, which identifies states significantly influencing the target decision policy. Tested on the 2-D Atari game Pong, our method achieved 100% knowledge transfer in identical tasks and either 100% knowledge transfer or a 30% reduction in training time for a rotated task, depending on the network architecture. In contrast, using standard Generative Adversarial Networks or Cycle Generative Adversarial Networks led to worse performance than training from scratch in the majority of cases. The results demonstrate that the proposed method ensured enhanced knowledge generalization in deep reinforcement learning.

INDEX TERMS Deep learning, Markov decision process, reinforcement learning, transfer learning, knowledge transfer.

NOTATION USED

s_t	State at the t-th time step.
r_t	Reward received at the t-th time step.
a_t	Action chosen at the t-th time step.
γ	Discount factor.
R	Reward function.
Q	Q-function.
$\mathcal{C}$	Correspondence function.
F	Environment model.
$\mathbf{K}$	Knowledge gained from a task.
$\mathbf{M}$	Experience memory.
G_S	Generator from source to target task.
G_T	Generator from target to source task.
$\mathcal{L}_{GAN}$	GAN loss.
$\mathcal{L}_{Cyc}$	Cycle-consistency loss.
$\mathcal{L}_Q$	Q-loss.
$\mathcal{L}_M$	Model loss.
λ_{Cyc}	Weight of cycle-consistency loss.
λ_Q	Weight of Q-loss.
λ_M	Weight of model loss.

ACRONYMS

DM	Decision making.
GAN	Generative Adversarial Network.
CycleGAN	Cycle-Consistent GAN.
MDP	Markov decision process
MuJoCo	Multi-Joint dynamics with Contact.
PlaNET	Deep Planning Network.
RL	Reinforcement learning.
TL	Transfer learning.
UNIT	Unsupervised Image-to-Image Translation Networks.

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

I. INTRODUCTION

The inherent ability of Reinforcement learning (RL) to dynamically learn complex policies through trial and error has shown great potential in solving diverse decision problems. Deep RL, which combines the advantages of RL with the power to handle high-dimensional data, has recently brought many advances. For instance, model-free methods have shown significant results in Multi-Joint dynamics with Contact (MuJoCo) environments [1], real-world robotic applications [2], and have demonstrated an ability to achieve super-human performance in Atari games [3], [4]. Model-based deep RL methods such as AlphaZero [5] and Deep Planning Network (PlaNET) [6] have also made significant progress. However, RL remains unsuitable for many real-world tasks, as errors can be extremely costly. One promising way to address this issue is through *Transfer learning (TL)* [7], [8], where skills and knowledge collected from similar tasks are applied to the current problem. Additionally, TL plays a crucial role in: developing agents capable of lifelong learning [9], multi-task learning [10], enabling simulation-to-real knowledge transfer in robotics [11], [12], [13], [14], [15], and advancing the development of general AI [16], [17].

Despite many advances, the use of transfer learning in RL, especially in deep RL, remains limited due to several challenges:

- *Weak ability of the RL agent to generalize to unobserved tasks*. For example, the output of a deep convolutional network for image data can be dramatically altered by a 1-pixel perturbation of the input image, [18]. This issue extends to RL, as image data often form the observable states in RL tasks. For instance, 1-pixel perturbations can lead to ineffective policies, [19]. RL methods frequently fail to reuse previously acquired knowledge even in similar tasks when the original image is rotated or when some colours are changed. It has also been shown that learning from scratch can be more efficient than fine-tuning a previously obtained model [20]. This significantly contrasts with the human ability to generalize and reuse previously acquired knowledge.
- *Challenging transfer of knowledge and experience* from previously solved tasks to unseen ones. A decision policy learnt from similar tasks may not always be effective in solving the current decision task. For example, optimal policies for driving a motorcycle in racing conditions are unsuitable and even dangerous for public roads.

The objective of this paper is to create an efficient method for one-to-one knowledge transfer between different RL tasks, with the aim of *improving* the agent's performance on the target task and *reducing* the training time required. The primary motivation is that any transferred object (skills or knowledge) is typically task- and policy-specific.[1] To ensure that the transferred skills are relevant and effective, we focus on identifying and transferring the most informative patterns. This approach enhances the RL agent's ability to generalize across tasks, thereby improving performance in the target task.

An additional, but important, reason why TL may fail is that the tasks involved can differ significantly in their dynamics and rewards. The proposed method addresses this by considering the matching of the dynamics of the involved RL tasks to ensure that appropriate behaviour patterns are transferred.

The proposed solution adopts a cyclic paradigm and is formulated as an *RL-specific modification of CycleGAN*. It introduces two new components to the loss function: model loss and Q-loss. The model loss captures the essential dynamic relationship between the involved RL tasks, while Q-loss prioritises states that affect learning the policy of the target RL task.

A. MAIN CONTRIBUTIONS OF THE PAPER

- **Introduction of an efficient method for knowledge transfer.** We propose a novel method for knowledge transfer between two different RL tasks based on an RL-specific modification of Cycle-Consistent Generative Adversarial Network (CycleGAN), designed to enhance generalization and reduce training time across tasks.
- **Establishment of a correspondence function.** We develop a correspondence function that learns and reveals the similarities between source and target RL tasks. This function plays a key role in facilitating efficient and accurate knowledge transfer.
- **Development of a four-component loss function.** Our four-component loss function incorporates model loss, Q-loss, and two additional components to better reflect task dynamics and account for the actual policy being used. This design improves the transfer of relevant knowledge and enhances the learning process.
- **Generalization of Generative Adversarial Network (GAN) and CycleGAN methods.** By introducing two new components to the loss function, we extend and generalize the GAN and CycleGAN methods. Our approach demonstrates that these standard techniques are special cases of our proposed framework, providing a broader, more powerful solution for knowledge transfer in RL.
- **Complete knowledge reuse in Pong tasks.** We achieve 100% knowledge reuse in experiments transferring between the original Pong and a *rotated* Pong environment, highlighting the method's ability to fully transfer learned skills without requiring re-training.
- **Handling of challenging tasks where standard methods fail.** Our method successfully handles tasks that are problematic for traditional GAN and CycleGAN methods, allowing for faster learning and improved performance where the only alternative would be to learn from scratch.

[1] The environment dynamics and rewards of RL tasks may differ.

1) ADVANTAGES OF THE PROPOSED METHOD

- **No reliance on paired data**: Unlike many transfer learning techniques, our approach does not require paired datasets, making it broadly applicable across various domains and tasks.
- **Task and domain independence**: The method is independent of the nature of the RL tasks involved, meaning it can be applied to diverse domains without task-specific manual engineering.
- **Flexible data formats**: It works with different data formats for states (e.g., images, sounds, numerical vectors), ensuring versatility and applicability to a wide range of applications. The method can be adapted to various RL tasks by selecting an appropriate network architecture tailored to the specific data format, not limited to image data alone.

The paper layout is as follows. Section II recalls the necessary background and formulates the considered TL problem. Section III constructs the correspondence function and proposes a novel method of its learning. Section IV describes the experimental evaluation of the proposed approach and compares it with baseline methods. Section V provides concluding remarks and outlines future research directions.

B. RELATED WORKS

Survey [8] systematically analyses recent advances in transfer learning for deep RL. Our research approach falls within the category of methods that employ mapping functions between the source and target tasks to facilitate knowledge transfer. A notable subset of this research focuses on learning shared features across RL tasks that are transferable. As demonstrated in [21], policies trained on intermediate-level features, referred to as *mid-level features*, exhibit superior generalization compared to policies trained directly on raw image observations. Work [22] leverages general features of two RL tasks with different dynamics. However, the method is based on paired image observations which are hard or impossible to obtain in practice. Work [23] achieved success in tasks differing in reward function by maintaining successor features and decoupling environment dynamic and reward function. Approach [24] introduces task similarity criterion and builds TL framework based on knowledge shaping, where for similar tasks, efficient transfer is theoretically guaranteed.

The pioneering work that used task correspondence was based on unsupervised image-to-image translation models CycleGAN, [25], and Unsupervised Image-to-Image Translation Networks (UNIT), [26]. Approach [20] achieved results on a specific set of tasks by finding correspondence between states of two RL tasks. The application potential of the approach is rather limited as problems like mode-collapse are present. Works [12] and [13] improved the approach proposed in [20] by introducing Q-function or object detection into the learning of the task correspondence. One of the recent approaches, [27], considers an environment model while learning the task correspondence, which is strongly inspired by the video-to-video translation model, [28].

II. BACKGROUND AND NOTATION

This section briefly recalls RL formalism and introduces the considered problem.

A. NOTATION

Throughout the text, sets are denoted by bold capital letters (e.g. $\mathbf{X}$), $\mathbb{N}$ and $\mathbb{R}$ are sets of natural and real numbers respectively. $\|x\|$ is the L1 norm of x. x_t is the value of x at discrete time $t \in \mathbb{N}$. $E_p[x]$ denotes the expected value of x with respect to a probability density p (if provided). Specific notations are provided at the beginning of the article.

We formalise the transfer problem in a general way by considering two RL tasks - the *source task*, S, and the *target task*, T, characterised by their respective task domains. $\mathbf{S}_S \times \mathbf{A}_S$ and $\mathbf{S}_T \times \mathbf{A}_T$, with $\mathbf{S}$ and $\mathbf{A}$ denoting a set of states and a set of actions respectively.

B. REINFORCEMENT LEARNING

Reinforcement learning (RL) considers an *agent* purposefully interacting with an *environment* by selecting actions. RL agent models its environment as *Markov decision process (MDP)*, [29] consisting of discrete sets of observable states $\mathbf{S}$ and actions $\mathbf{A}$. Set $\mathbf{S} \times \mathbf{A}$ is referred to as the *task domain*. At each time t, the agent observes environment state $s_t \in S$ and takes action $a_t \in \mathbf{A}$. Executing action a_t at state s_t: i) causes a transition of the environment to state s_{t+1} according to *transition function* that describes $p(s_{t+1}|s_t, a_t)$, and ii) provides reward r_t, i. e. the value of reward function $R(s_{t+1}, a_t, s_t) : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \mapsto \mathbb{R}$. The agent's *goal* is to learn policy $\pi : \mathbf{S} \mapsto \mathbf{A}$ that maximises the accumulated reward.

The solution of a MDP task is the *optimal* policy π^*:

$$\pi^* = \arg\max_{\pi \in \mathbf{\Pi}} E\left[\sum_{t=1}^{N} \gamma^t R(s_{t+1}, \pi(s_t), s_t)\right], \quad \text{where } \mathbf{\Pi} = \{\pi(s_t)\}_{t=1}^{N} \tag{1}$$

with *decision horizon* $N \in \mathbb{N} \cup \{\infty\}$ and *discount factor* $\gamma \in (0, 1)$. The RL agent learns to act optimally within MDP when the transition function and reward function are unknown. A good RL modifies π over time to gradually get it closer to an optimal policy. Q-learning, a model-free RL algorithm, is one of the traditional solution approaches. It aims to learn Q-function (aka *state-action-value function*) that quantifies the expected value of future discounted reward over the states induced by π^* for given starting state s and action a.

$$Q(s, a) = E_{\pi^*}\left[\sum_{t=1}^{N} \gamma^t R_t(s_{t+1}, \pi^*(s_t), s_t) \middle| s_1 = s, a_1 = a\right]. \tag{2}$$

Discount factor γ expresses the agent's preferences towards immediate reward over future ones. The estimate of (2),

$\hat{Q}(s, a)$, can be gradually learnt on the stream of data records (s_t, a_t, r_t, s_{t+1}) using for instance, temporal difference learning, [30]:

$$\hat{Q}_{t+1}(s_t, a_t) = (1-\alpha)\hat{Q}_t(s_t, a_t) + \alpha(r_t + \gamma \max_{a \in a} \hat{Q}_t(s_{t+1}, a)), \tag{3}$$

where $\alpha \in (0, 1)$ is a parameter called *learning rate* and $r_t = R(s_{t+1}, a_t, s_t)$. The learning starts with an initial estimate of the Q-function, $\hat{Q}_0(s, a)$. The learned, approximately optimal, decision rule is then

$$\pi^* \left(s \mid \hat{Q} \right) = \operatorname*{argmax}_{a \in a} \hat{Q}(s, a). \tag{4}$$

C. DEEP Q-LEARNING

Whenever the state space is huge, for instance, when the state is given by a video frame, efficient learning of Q-function calls for numerical approximation. The state-of-the-art in function approximation points to deep neural networks (DNN) as a suitable methodology, [31].

Deep Q-networks (DQN), [32], use a standard off-policy Q-learning, [30], and DNN to estimate the Q-function (2).

DQN approximates Q-function by a deep neural network with parameters that can be trained similarly to the supervised learning, [3]. However, the supervised learning assumes i. i. d. input data. Moreover, output values are expected to be the same for the same inputs, [33]. Neither of these assumptions is met in RL tasks. The consecutive states are usually highly correlated (e. g. video frames) and thus very far from being i. i. d. Output values also contain learned Q-function that evolves during learning. This makes the learning process unstable. To enable data reuse and stabilise the learning, DQN uses an *experience replay technique* to remove correlations in the observed sequence and employs an additional *target network*[2] to stabilise the output values, see [3] for details.

Experience replay technique considers that the last n_M data records (so-called *experience memory*, denoted as $\mathbf{M}$) are stored in a memory buffer. At each learning step, a mini-batch of length $n_B \in \mathbb{N}$ is randomly sampled from the memory buffer and is used to update the neural network that approximates Q-function. It brings the learning data closer to being i. i. d.

Target network is an additional network[3] serving for stabilising the learning. The idea is as follows. The parameters of the *original network* are updated at every learning step, while *target network* is used to retrieve output values and stays static, i.e. its parameters do not change. Every $n_U \in \mathbb{N}$ steps, the original and target networks are synchronised. Details on the DQN algorithm, see Appendix.

D. CYCLE-CONSISTENT GAN

CycleGAN, [25], is based on GAN, [34], and was originally proposed for image-to-image translation. The idea behind *cycle consistency* is that data that has been translated to a new domain and then recovered from it, should not change.

CycleGAN operates with two mappings G_S and G_T called *generators*[4]

$$G_S : \mathbf{S}_S \to \mathbf{S}_T \quad \text{and} \quad G_T : \mathbf{S}_T \to \mathbf{S}_S. \tag{5}$$

They are learnt as two GANs, that is, simultaneously with the corresponding discriminators D_S and D_T. Generators learn to map states from $\mathbf{S}_S$ to $\mathbf{S}_T$ and vice-versa, while discriminators learn to *distinguish* a real state from a state mapped by a generator. Mappings G_S, G_T, D_S and D_T are constructed as neural networks with their architecture depending on the data format. For instance if states are images, convolutional layers are often used.

Learning in CycleGAN minimises a two-component loss. The first is *adversarial loss*, $\mathcal{L}_{GAN}$ comes from GAN and is given by

$$\begin{aligned} &\mathcal{L}_{GAN} = \\ &E_{s_S}\left[\log D_S(s_S)\right] + E_{s_T}\left[\log\left(1 - D_S\left(G_T\left(s_T\right)\right)\right)\right] \\ &\quad + E_{s_T}\left[\log D_T(s_T)\right] + E_{s_S}\left[\log\left(1 - D_T\left(G_S\left(s_S\right)\right)\right)\right] \end{aligned} \tag{6}$$

The adversarial training encourages mappings G_S and G_T (5) to produce outputs indistinguishable from the real ones, i. e. respective sets $\mathbf{S}_S$ and $\mathbf{S}_T$. However, minimising $\mathcal{L}_{GAN}$ does not prevent the network from mapping the same set of input images to any permutation of images in the target domain.

The second component is *cycle-consistency* loss, $\mathcal{L}_{Cyc}$, that has the following form:

$$\begin{aligned} \mathcal{L}_{Cyc} = &E_{s_S}\left[\left\|G_T\left(G_S\left(s_S\right)\right) - s_S\right\|\right] \\ &+ E_{s_T}\left[\left\|G_S\left(G_T\left(s_T\right)\right) - s_T\right\|\right]. \end{aligned} \tag{7}$$

Minimisation of cycle-consistency loss $\mathcal{L}_{Cyc}$ ensures that every state $s_S \in \mathbf{S}_S$ must be recoverable after mapping it back to $\mathbf{S}_T$, i.e. $G_T(G_S(s_S)) \approx s_S$. The same requirement applies to every state $s_T \in \mathbf{S}_T$.

III. TRANSFER LEARNING FOR RL

Humans have a remarkable ability to generalise. They do not learn everything from scratch but rather reuse earlier acquired knowledge to a new task or domain.[5] Generally, finding common patterns between different tasks and effectively transferring the concepts learned from one task to another is an essential characteristic of high-level intelligence. Thus, the efficient solution of transfer learning will allow for the creation of intelligent agents that can mimic human thinking and solve problems in a much more explainable way. Moreover, efficient reusing the acquired knowledge may accelerate the learning process and make complex tasks learnable.

[2]Note that name *target network* in DQN generally does not refer to *target task*.

[3]That has the same architecture as the original network.

[4]That translate data between source and target domains.

[5]Developmental psychologists have shown that as early as 18 months old, children can infer intentions and imitate the behaviour of adults, [35]. The imitation is complex as children must infer a match between their observations and internal representations, effectively linking the two diverse domains.

This section, we formalises a problem of transfer learning between two RL tasks, empirically introduces a correspondence function reflecting the similarity of two RL tasks and proposes an RL-specific modification of CycleGAN algorithm that realises knowledge transfer between two RL tasks. The proposed transfer i) considers behaviours, which are most useful for the target task; ii) captures and respects common patterns in transition dynamics of the involved RL tasks.

A. PROBLEM FORMULATION

We consider two RL tasks: the *source task*, S, and the *target task*, T with their respective task domains $\mathbf{S}_S \times \mathbf{A}_S$ and $\mathbf{S}_T \times \mathbf{A}_T$. Each of the tasks corresponds to MDP with its own environmental dynamics and reward function, see Section II-B. Transition functions of the tasks as well as theirs reward functions may be different.

Intuitively, the success of transfer between two RL tasks depends on the degree of similarity between these tasks. If the tasks are dissimilar, the transfer of inappropriate knowledge may significantly worsen the resulting performance in the target task. Therefore, the success of the transfer broadly depends on the existence of some common properties between the source and target tasks. The similarity can be perceived from various perspectives, such as sharing the same environment, obeying similar laws of physics, or involving similar objects for interaction. For instance, when driving a motorcycle, encountering an animal on the road may correspond to pulling the brake levers, just as when driving a car, the sight of a person crossing the road can lead to pressing the brake pedal.

This work uses an abstract notion of similarity, inspired by human learning when tackling related problems. Two tasks are similar if they share some common properties, and the knowledge acquired in one task proves to be beneficial in solving the other. This empirical definition can be more formally introduced as follows.

Definition 3.1 (Correspondence function): Consider source S and target T tasks with respective domains $\mathbf{S}_S \times \mathbf{A}_S$ and $\mathbf{S}_T \times \mathbf{A}_T$. A *correspondence function*, $\mathcal{C} : (\mathbf{S}_T \times \mathbf{A}_T) \mapsto (\mathbf{S}_S \times \mathbf{A}_S)$, is a mapping, which reveals the similarity of the involved RL tasks in terms of the dynamics of the tasks' environments and the associated Q-functions.

It is clear that function $\mathcal{C}$ establishes the relationship between similar patterns in behaviour of the target and source tasks that are necessary for knowledge transfer. So, if Q_S is the optimal Q-function for the source task, then Q-function

$$Q_S(\mathcal{C}(.,.)) : \mathrm{S}_T \times \mathrm{A}_T \mapsto \mathrm{R} \tag{8}$$

gives better performance[6] on the target task than a random policy.

Let us assume (for brevity) that the action spaces of the source and the target RL task are identical, i.e. $\mathbf{A}_S = \mathbf{A}_T$. Let mutually corresponding actions be found using identity mapping regardless of the current state.[7] Thus, we need to learn a mapping indicating corresponding states, i. e. the correspondence function for states. The searched correspondence function $\mathcal{C}$ is then obtained as follows:

$$\mathcal{C}(s_T, a_T) = (G_T(s_T), I(a_T)), \qquad \forall (s_T, a_T) \in \mathrm{S}_T \times \mathrm{A}_T, \tag{9}$$

where G_T is the generator from (5) mapping states from the *target task* to states from the *source task* and $I(.)$ is an identity mapping.

The correspondence function is unknown to RL agent and the next section describes how to learn it.

B. LEARNING OF CORRESPONDENCE FUNCTION

The proposed learning is inspired by CycleGAN, see Section II-D, where the learning minimises a discriminative *loss function*, which makes the similarity metric small for similar patterns and large otherwise. Even direct application of CycleGAN to the states brought some success in policy transfer, see for instance [20]. However, data records in experience memories comprise richer yet unused information that may be helpful for the transfer of knowledge. We propose to include additional components into the loss function minimised in CycleGAN learning. They will consider unused information and make the learned correspondence entirely relevant to RL. The proposed loss will ensure that the learnt function $\mathcal{C}$ captures all patterns significant for the intended TL. In particular, the loss should respect both the dynamics and Q-function of the source task.

This work proposes adding two new components to the CycleGAN losses, (6), (7):

- Q-loss $\mathcal{L}_Q$ - a loss that reflects how the Q-function learned from the source task, Q_S, copes with impreciseness in learned generators G_T and G_S.
- Model-loss $\mathcal{L}_M$ - a loss that reflects the influence of the environment model of the source task.

Let us explain the reasons for introducing the new components and their forms.

1) Q–LOSS

The Q-function, Q_S, plays a central role in RL as it defines the optimal policy for the source task. When transferring knowledge from a source task S to a target task T, it is essential to preserve the Q-function's accuracy for states relevant to the decision making (DM) process. This motivates the introduction of Q-loss, $\mathcal{L}_Q$, which ensures that the learned correspondence function, $\mathcal{C}$, maintains consistency between the value estimates of corresponding states in both domains.

The *cycle-consistency* loss (7) ensures that the generators G_S and G_T map between the source and target domains in a consistent way. However, cycle-consistency alone does not prioritize the states that are most critical for decision-making

[6] Performance is measured by average reward per time.

[7] More specifically, all actions of the source and target task have the same labels and meanings (e.g. $a = 1$ stands for "up"). Therefore, no mapping between source and target task action spaces is necessary.

in RL. The Q-loss directly incorporates the Q-function, encouraging the generators to focus on the states that matter most for choosing optimal actions. Mathematically, the Q-loss is defined as:

$$\mathcal{L}_Q = E_{a,s_S}\left[\left\|Q_S\left(G_T\left(G_S\left(s_S\right)\right), a\right) - Q_S\left(s_S, a\right)\right\|\right] \quad (10)$$

This loss minimizes the difference between the Q-function values of state s_S in the source task and its mapped counterpart after a round-trip through the generators ($G_T(G_S(s_S))$). In simpler terms, this forces the correspondence function $\mathcal{C}$ to retain the critical information from the states in the source domain that is essential for determining the optimal policy, ensuring that this information is preserved after mapping between tasks S and T.

The rationale behind this is that for effective knowledge transfer in RL, it is not enough for the state representations to be similar visually or structurally; they must also be similar in terms of their impact on decision-making, as captured by the Q-function. By focusing on states that are important for action selection, the Q-loss makes the correspondence function more suitable for transferring policies between tasks.

2) MODEL LOSS

In reinforcement learning, tasks are inherently dynamic, meaning that a state's importance often depends on the actions taken and how the state evolves over time. This dynamic nature introduces a key challenge for transferring knowledge between tasks, as it is not just the individual states that matter but the transitions between them. To address this, we introduce the *model loss*, $\mathcal{L}_M$, which ensures that the correspondence function respects the underlying dynamics of the source and target tasks.

While losses like $\mathcal{L}_{GAN}$, $\mathcal{L}_{Cyc}$, and $\mathcal{L}_Q$ focus on individual state mappings, they do not ensure that the temporal dynamics of the target task align with those of the source task. In other words, even if individual states match, the transitions between states (due to actions taken) might not be consistent. The model loss addresses this by incorporating the environment model F_S of the source task. The model F_S predicts the next state based on the current state and action:

$$F_S : (s_t, a_t) \mapsto s_{t+1} \quad (11)$$

To ensure that the learned correspondence function, $\mathcal{C}$, captures the dynamic relationships between the source and target tasks, we define the model loss $\mathcal{L}_M$ as:

$$\mathcal{L}_M = E_{s_{Tt}, a_{Tt}, s_{Tt+1}}\left[\left\|F_S\left(G_T\left(s_{Tt}\right), a_{Tt}\right) - G_T\left(s_{Tt+1}\right)\right\|\right] \quad (12)$$

This loss ensures that the transitions in the target task are consistent with those in the source task when mapped through the correspondence function. Specifically, if an action a_{Tt} taken in the target task leads to a state transition from s_{Tt} to s_{Tt+1}, the model loss ensures that this transition corresponds to a valid transition in the source task. In other words, applying a_{Tt} to the mapped state $G_T(s_{Tt})$ should lead to a state $G_T(s_{Tt+1})$ that is predicted by the source task's environment model F_S.

Intuitively, this means that the correspondence function not only matches individual states between the source and target tasks but also ensures that the way states evolve over time (due to actions) is consistent. This is crucial for transferring knowledge about dynamic tasks, where the sequence of states and actions is key to solving the problem.

In summary, the model loss ensures that the correspondence function respects the temporal dynamics of both the source and target tasks, making it suitable for transferring policies between dynamic RL tasks. Together, the Q-loss and model loss guarantee that the transferred knowledge is useful both for individual states and for the dynamic relationships between them.

3) TOTAL LOSS

The proposed total loss comprises all the components (6), (7), (10) and (12) and, thus, has the following form:

$$\mathcal{L} = \mathcal{L}_{GAN} + \lambda_{Cyc}\mathcal{L}_{Cyc} + \lambda_Q\mathcal{L}_Q + \lambda_M\mathcal{L}_M, \quad (13)$$

where λ_{Cyc}, λ_Q and λ_M are *loss parameters* that define relative influence (weight) of the respective components.

The proposed approach, which minimises 4-component loss (13), generalises GAN, [34], and CycleGAN, [25], methods often used for transfer learning. It is easy to see that GAN and CycleGAN can be obtained by setting some of parameters λ_Q, λ_M, λ_{Cyc} in (13) to zeros as follows:

- $\lambda_Q = \lambda_M = \lambda_{Cyc} = 0$ (for GAN),
- $\lambda_Q = \lambda_M = 0$ (for CycleGAN).

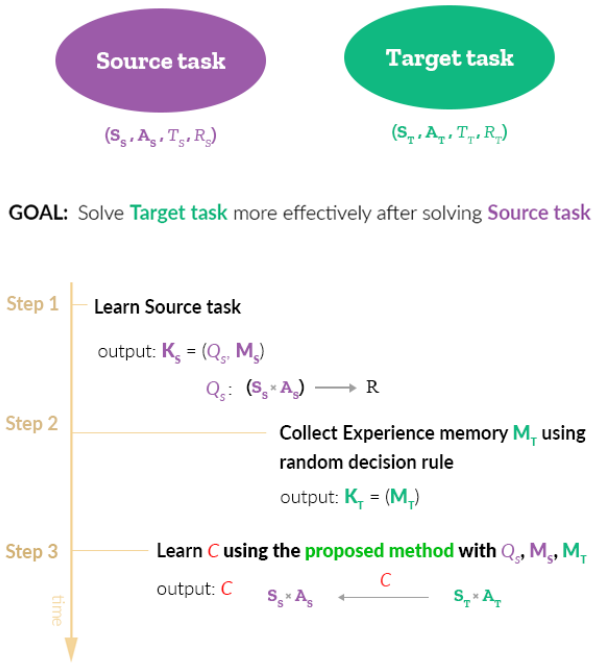


FIGURE 1. The proposed TL between tasks S and T.

C. TRANSFER LEARNING: ALGORITHM

The main steps of the proposed algorithm:

Step 1 The agent first solves task S by the DQN algorithm (see Section II-C). The obtained knowledge, $\mathbf{K}_S = (Q_S, \mathbf{M}_S)$, consists of learned Q-function, Q_S, and collected experience memory $\mathbf{M}_S = \left((s_t, a_t, s_{t+1}, r_t)_{i=1}^{n_M}\right)$.

Step 2 The agent applies a random decision rule to task T, collects experience memory $\mathbf{M}_T$. Further the agent uses $\mathbf{M}_T$ together with knowledge $\mathbf{K}_S$, to solve *target task* T more efficiently, see Figure 1.

Step 3 The assumed similarity of the tasks S and T guarantees the existence of correspondence function $\mathcal{C}$ (see Definition 3.1). The agent uses knowledge $\mathbf{K}_S = (Q_S, \mathbf{M}_S)$ and memory $\mathbf{M}_T$ to learn correspondence function $\mathcal{C}$. Hence the correspondence function is used to transform state-action pairs from the target task to the source task.

Step 4 Existence of a correspondence function $\mathcal{C}$, allows to express Q-function of the target task, Q_T, via Q-function of the source task, Q_S, and learnt correspondence function $\mathcal{C}$ as follows:

$$Q_T(s_T, a_T) = Q_S(\mathcal{C}(s_T, a_T)), \ \forall (s_T, a_T) \in (S_T \times A_T)\,. \tag{14}$$

Then the agent can use Q-function Q_S of the source task to choose the optimal actions in the target task.

1) NOTE ON IMPLEMENTATION

Similarly to GAN, in the considered case of TL we have two experience memories with *mutually unpaired* entries: $\mathbf{M}_S$ for the source task and $\mathbf{M}_T$ for the target task. The proposed algorithm learns the correspondence function, $\mathcal{C}$, that will match them.

The experience memory $\mathbf{M}_S$ is obtained as a by-product of DQN algorithm used for learning the optimal Q-function Q_S. However, the proposed method does not strictly require usage of DQN. It is important that it can be applied to any algorithm giving $\mathbf{M}_S$ and Q_S (where Q_S is a differentiable function).

2) NOTE ON THE USE

This paper considers using the transfer learning method just once, in the beginning of interaction with the target task. Other ways, however, might be explored such as when partially optimal strategy is found to further improve it.

IV. EXPERIMENTAL PART

To test the efficiency of the proposed approach, two experiments on the Atari game Pong, [36], were conducted. The performance of the approach was evaluated based on an average accumulated reward per game. GAN and CycleGAN were used as baseline methods.

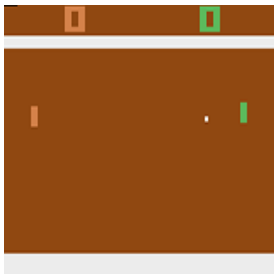


FIGURE 2. Standard pong, [36].

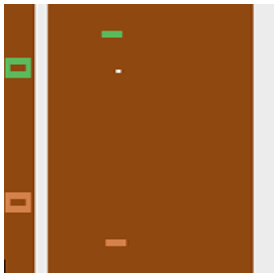


FIGURE 3. Pong rotated by 90 degrees, [36].

A. DOMAIN DESCRIPTION

Pong is a two-dimensional game simulating table tennis. There are six available actions ('do nothing', 'fire', 'move up', 'move down', 'move up fast', 'move down fast'). The last four observed image frames served as a task state. The agent learned to play the game using the DQN algorithm, Section II-C, and, thus, learned the Q-function. To test the approach described in Section III-B, the agent also learned environment model F.

B. EXPERIMENT DESCRIPTION AND SETUP

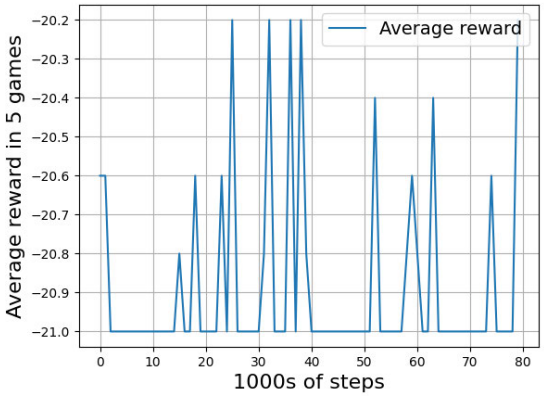
The proposed TL method was tested in two experiments.

Experiment 1: The source and target tasks were the same, i.e. game Pong (screenshot is shown in Figure 2). The main aim of this experiment was to verify the ability of the proposed approach to find the identity transformation.

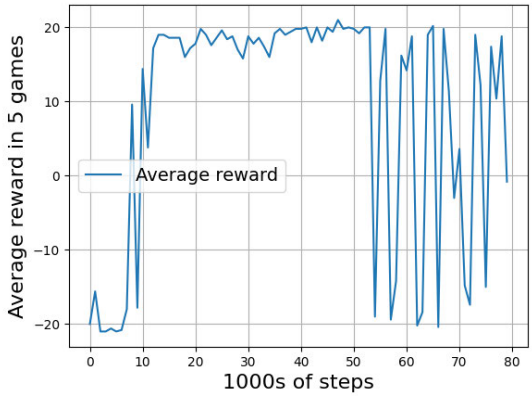
Experiment 2: The source task was the original Pong while the target task was rotated Pong (see screenshot in Figure 3). The game remained the same, but all image frames were rotated by 90 degrees.

Each experiment consists of the following steps:

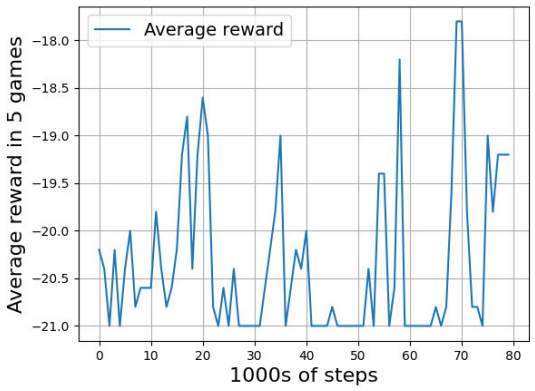
1) The agent played the *source task* (standard Pong), learned the optimal policy by DQN and obtained the optimal Q-function Q_S, environment model F and experience memory $\mathbf{M}_S$ containing 10000 data entries collected at the end of the game.
2) The agent played the *target task* (standard Pong in Experiment 1 or rotated Pong in Experiment 2) using random policy and obtained data for experience memory $\mathbf{M}_T$ containing 10000 data entries.
3) The agent started learning the correspondence function $\mathcal{C}$ using the method from Section III with the



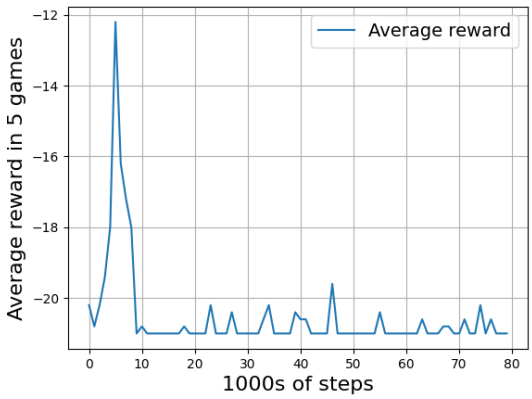
a) **GAN**
$\lambda_{Cyc} = \lambda_Q = \lambda_M = 0$



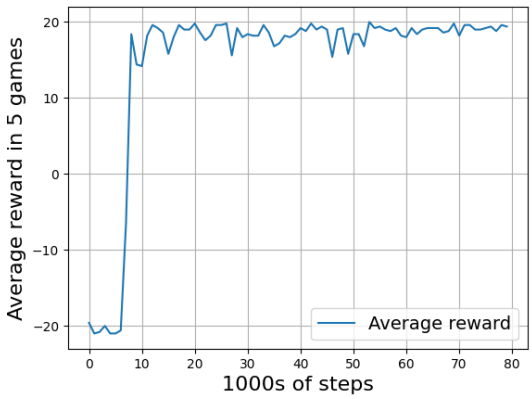
b) **CycleGAN**
$\lambda_{Cyc} = 10, \lambda_Q = \lambda_M = 0$



c) Loss (13) with $\lambda_{Cyc} = 0, \lambda_Q = 1, \lambda_M = 0$



d) Loss (13) with $\lambda_{Cyc} = 0, \lambda_Q = 0, \lambda_M = 10$

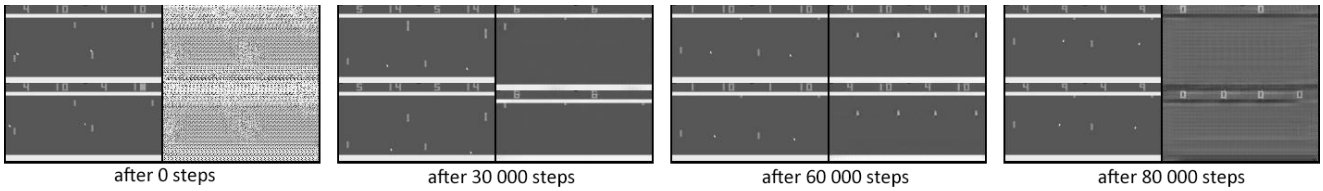


e) Loss (13) with $\lambda_{Cyc} = 1, \lambda_Q = 1, \lambda_M = 1$

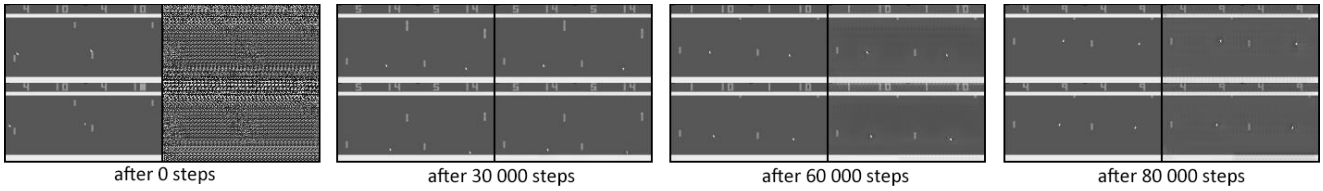
FIGURE 4. Experiment 1: Average accumulated reward per game when playing five games with the transformed *Q*-function (14). The agent paused the correspondence function learning each 1000 learning steps and played five games where the average reward gained per game is displayed. The performance is shown for different values of loss parameters λ_{Cyc}, λ_Q and λ_M. Figure 4a and 4b show the baselines using *GAN* and *CycleGAN* methods.

Q-function Q_S, environment model F and experience memories $\mathbf{M}_S$ and $\mathbf{M}_T$,

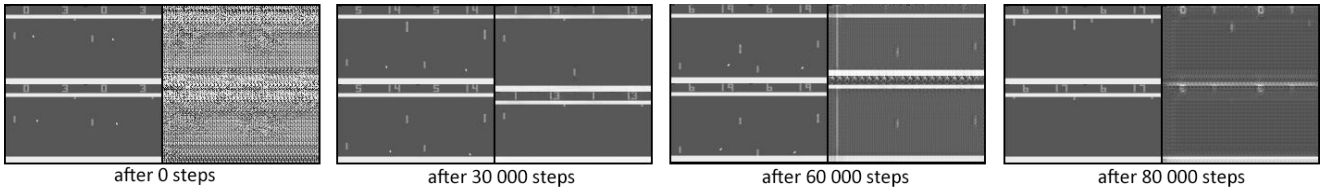
4) For every 1000 learning steps, the agent:
   - suspends learning of correspondence function $\mathcal{C}$,



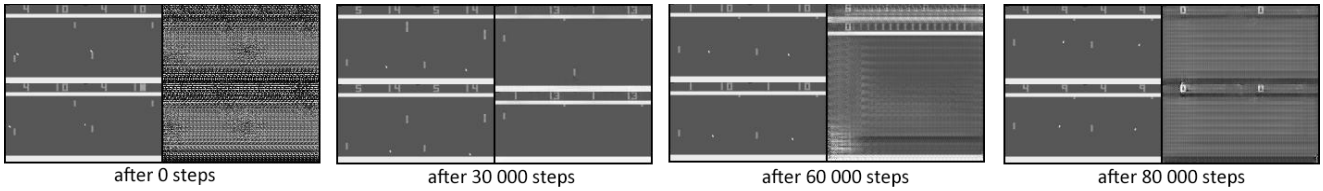
a) **GAN** $\lambda_{Cyc} = 0, \lambda_Q = 0, \lambda_M = 0$



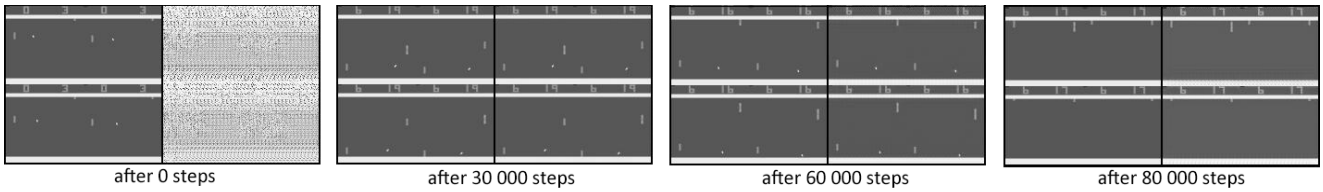
b) **CycleGAN** $\lambda_{Cyc} = 10, \lambda_Q = 0, \lambda_M = 0$



c) $\lambda_{Cyc} = 0, \lambda_Q = 1, \lambda_M = 0$



d) $\lambda_{Cyc} = 0, \lambda_Q = 0, \lambda_M = 10$



e) $\lambda_{Cyc} = 1, \lambda_Q = 1, \lambda_M = 1$

FIGURE 5. Experiment 1: Screenshots of the game depicting the progress of learning correspondence function $\mathcal{C}$, (9), after 0, 30000, 60000 and 80000 steps. The results are shown for different values of parameters λ_{Cyc}, λ_Q and λ_M (13). The left parts are game frames of the *target* task serving as states, and the right parts are the same states mapped by the learned correspondence function, $\mathcal{C}$.

- uses learnt $\mathcal{C}$ and the Q-function transformed from the *source task*, see (14), to play five games of the *target task*, and
- computes the average accumulated reward per game.

5) The agent played the *target task* while using the learned correspondence[8] and Q-function Q_S transferred from the *source* task. At the same time the agent uses DQN and fixed $\mathcal{C}$ to continuously fine-tune Q-function Q_T of the target task.

The *key metric* to evaluate the success of the knowledge transfer was the average accumulated reward per game.

Baseline methods: The results are compared with two baselines—using GAN and CycleGAN methods [25], [34], which have recently been applied for knowledge transfer in similar settings [20]. Experiment 2 also includes *fine-tuning*

[8]The correspondence function that achieved the highest average accumulated reward per game in the previous step was used here.

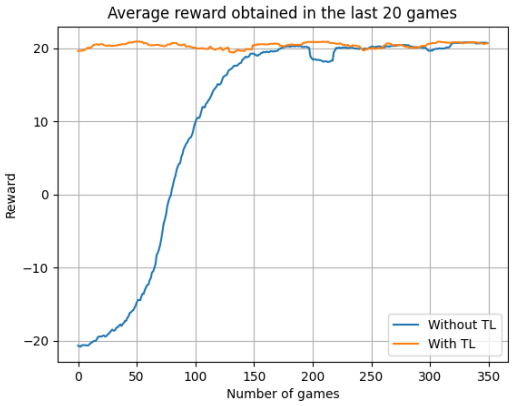


FIGURE 6. **Moving average of reward per game computed from the last 20 games depending on the number of *Pong* games played. The blue line denotes learning from scratch, i. e. without TL. The orange line denotes the case with TL, i.e. when the agent learns the correspondence function and uses the transformed *Q*-function (14). The *Q*-function Q_T is continuously learned during the game in both cases.**

the Q-function from the source task as a baseline, as it is a commonly used transfer learning method.

The following sections provide the key details of the experiments performed and their results.

C. EXPERIMENT 1

This experiment aimed to test transfer learning when *source* and *target* tasks are identical.

G_S and G_T generators (see Section III-B) were constructed as neural networks with convolutional layers. Their specific architecture was taken from [37]. The discriminators D_S and D_T were also constructed as neural networks with convolutional layers with the architecture as in [38].

The parameters of all of the networks were initialized from Gaussian distribution $N(0, 0.02)$. The transfer learning with the loss (13) was tested for all the combinations of the parameters: $\lambda_{Cyc} \in \{0, 1, 10\}$, $\lambda_Q \in \{0, 1\}$ and $\lambda_M \in \{0, 1, 10\}$.

1) RESULTS

The results presented in Figure 4 - Figure 6 highlight the effectiveness of the proposed method in comparison to baseline models. After every 1000 learning steps, the agent pauses to play five games, and the average reward per game is recorded.

In Figure 4, the best results are observed when all loss components ($\mathcal{L}_{Cyc}$, $\mathcal{L}_Q$, and $\mathcal{L}_M$) are included in the total loss function (13) with parameters $\lambda_{Cyc} = \lambda_Q = \lambda_M = 1$, as shown in Figure 4e. This configuration achieves nearly the maximum reward (21), indicating that the method transfers knowledge effectively and optimizes performance.

The significance of the new components is demonstrated by Figure 4c and Figure4d. When only one of the new components ($\mathcal{L}_Q$ or $\mathcal{L}_M$) is included, the performance drops noticeably. This result emphasizes that both components are critical to achieving successful knowledge transfer. Other parameter combinations did not yield meaningful results and are therefore not presented here.

In contrast, the baseline methods perform poorly. The GAN baseline (Figure 4a) fails to yield meaningful results, while the CycleGAN baseline (Figure 4b) shows initial success, but its performance quickly becomes unstable, indicating that it cannot maintain an effective correspondence function over time. The fluctuations observed in the CycleGAN curve in Figure 4b stem from the adversarial nature of the training process, and adding the proposed losses appears to help mitigate this instability.

Figure 4 visually demonstrates the correspondence between the source and target tasks, confirming that the best performance is obtained when all components of the loss function are active. Although the CycleGAN baseline shows some visual accuracy, its inconsistency is evident in the unstable reward progression.

Finally, Figure 6 compares the performance of an agent learning from scratch with one that transfers knowledge using the proposed method. The agent that reuses previously learned knowledge reaches high performance almost immediately, while the agent learning from scratch requires much more time to reach the same level. This highlights the efficiency of our method both in transferring knowledge and reducing training time.

D. EXPERIMENT 2

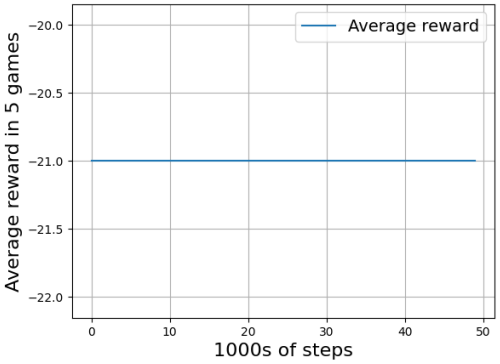
In Experiment 2, the *target task* is the original Pong with image frames rotated by 90 degrees (see Figure 3).

Generators G_S and G_T, (see (5) and Section III-B) are constructed as neural networks. Two types of generators were used in the experiment. The architecture of the first one, referred to here as the **resnet generator**, was taken from [37] and then followed by a rotation layer, see [39]. The second type, referred to as the **rotation generator**, was composed of the mentioned rotation layer only. Discriminators D_S and D_T are constructed by neural networks with convolutional layers with the architecture as in [38].

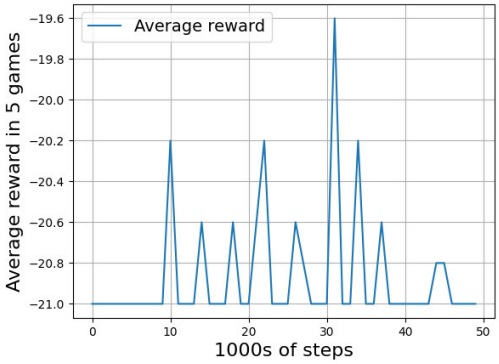
1) RESULTS

The proposed approach was tested with various values of the loss parameters λ_{Cyc}, λ_Q, and λ_M (from (13)). Figure 7 - Figure 9 present the best-achieved performance of our method, compared to baseline methods.

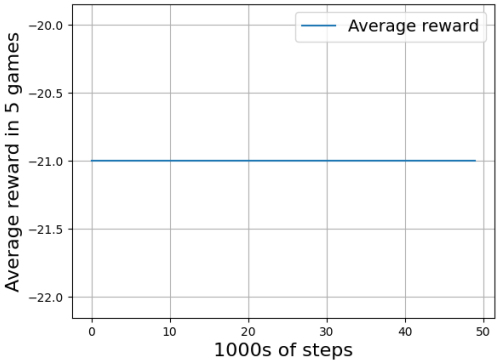
Figure 7 depicts the average reward per game over five games. Similar to Experiment 1, after every 1000 learning steps, the agent pauses the learning of the correspondence function and plays five games of the target task. The results show that the *rotation* generator achieves nearly perfect knowledge transfer, with rewards approaching the maximum score of 21. This indicates that the *rotation* generator can establish an effective correspondence between the source and target tasks, enabling high-performance transfer.



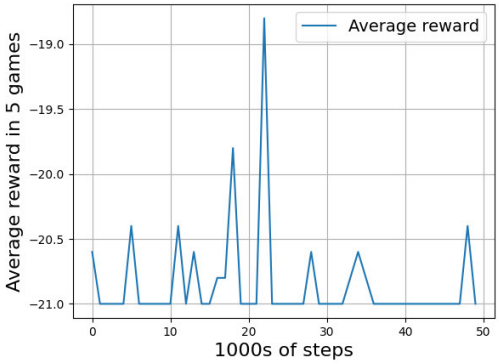
a) **Rotation generator** using **GAN**



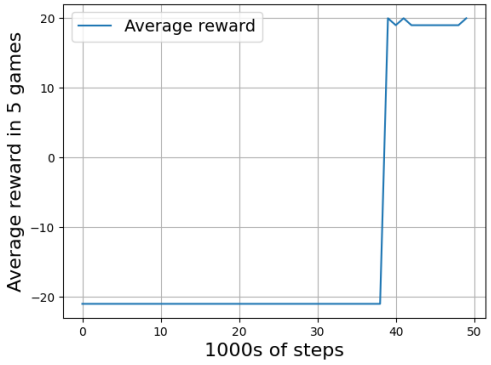
b) **Resnet generator** using **GAN**



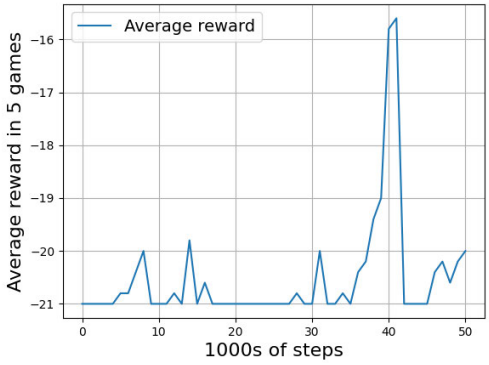
c) **Rotation generator** using **CycleGAN**



d) **Resnet generator** using **CycleGAN**



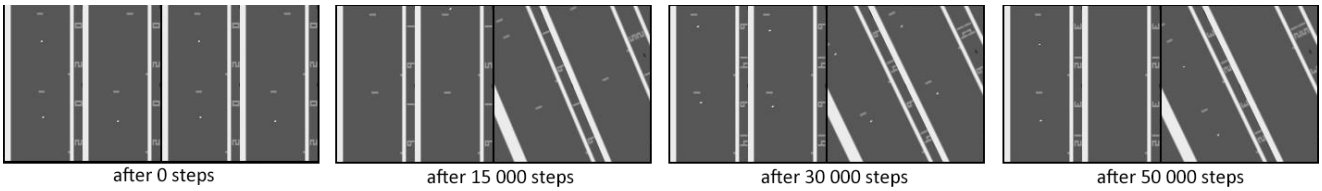
e) **Rotation generator** with $\lambda_{Cyc} = \lambda_Q = 0, \lambda_M = 10$



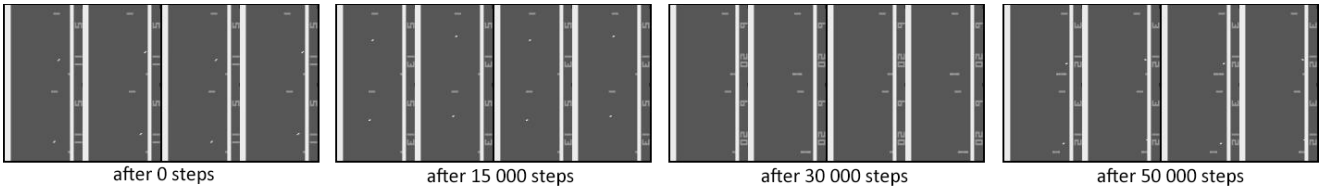
f) **Resnet generator** with $\lambda_{Cyc} = 1, \lambda_Q = 0, \lambda_M = 1$

FIGURE 7. **Average accumulated reward in five games when playing Rotated Pong with the transformed *Q*-function (14). The agent paused the correspondence function learning each 1000 learning steps and played five games where the average reward gained per game is displayed. The results are shown for the rotation and the resnet generator with the best settings of the loss parameters in each case (e, f) as well as with using GAN and CycleGAN baselines (a-d).**

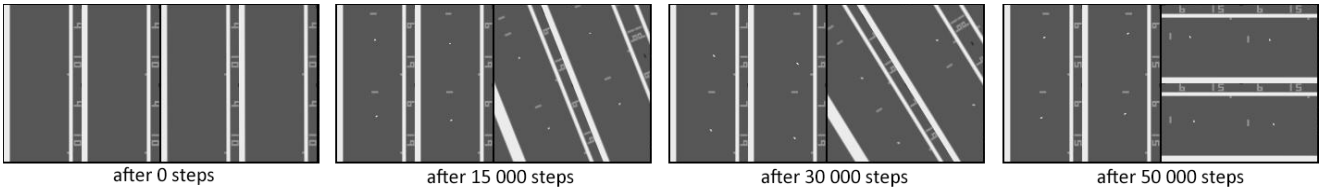
In contrast, although the *resnet* generator did not yield perfect results, it still learned a reasonable correspondence function, particularly after 50,000 steps, as shown in Figure 8. This partially successful correspondence was then used to



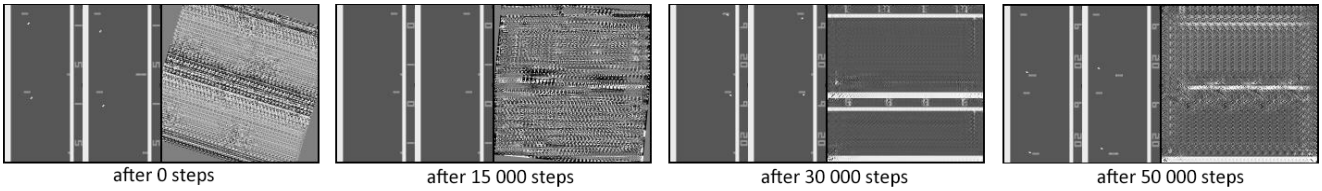
a) Rotation generator using **GAN**



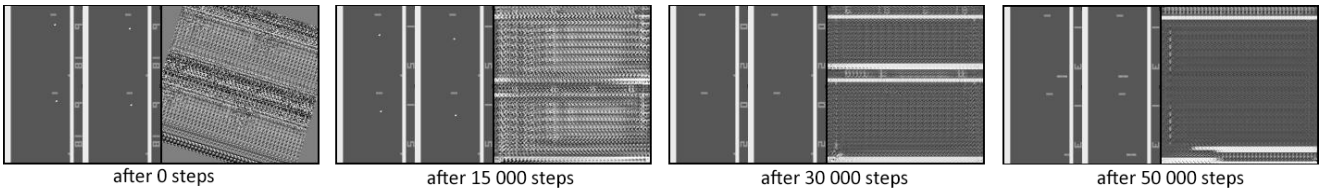
b) Rotation generator using **CycleGAN**



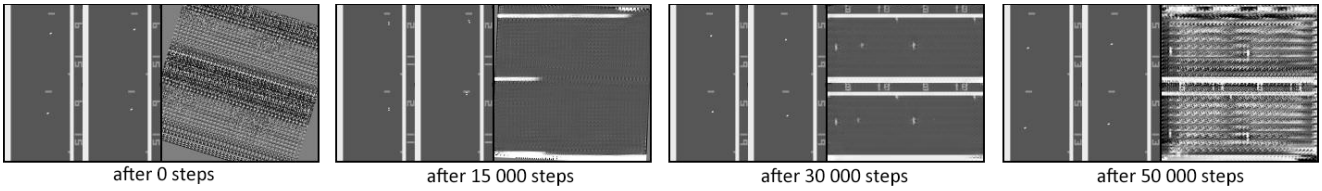
c) Rotation generator with $\lambda_{Cyc} = 0$, $\lambda_Q = 0$, $\lambda_M = 10$



d) Resnet generator using **GAN**



e) Resnet generator using **CycleGAN**



f) Resnet generator with $\lambda_{Cyc} = 1$, $\lambda_Q = 0$, $\lambda_M = 1$

FIGURE 8. **Experiment 2: Screenshots of the game depicting progress in learning the correspondence function $\mathcal{C}$ (9) after 0, 15000, 30000 and 50000 steps. The results are shown for the rotation and the resnet generators with the best settings of parameters λ_{Cyc}, λ_Q and λ_M (13) as well as with using GAN and CycleGAN baselines. The left parts of the pictures are game frames of the *target* task representing the states, and the right parts are the same states transformed by the correspondence function $\mathcal{C}$.**

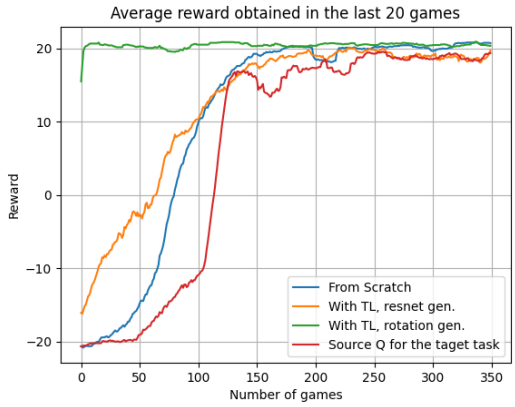


FIGURE 9. **Moving average of reward per game computed from the last 20 games depending on the number of played games for the game rotated Pong for four different agents - an agent learning the game from scratch (blue line), an agent using the correspondence function learned with the resnet generator (orange line), an agent using the correspondence function learned with the rotation generator (green line) and an agent reusing only the *Q*-function without any correspondence function (red line). The agents were continuously learning the *Q*-function.**

fine-tune the Q-function for the target task. Importantly, this fine-tuning process led to much better results than training the Q-function from scratch, demonstrating that even an imperfect correspondence can significantly accelerate learning. G The baseline methods, using standard GAN and CycleGAN, failed to produce any usable correspondence for knowledge transfer. This is clearly illustrated in Figure 7a and Figure 7b, where the agent's poor performance highlights the limitations of these methods for reinforcement learning tasks.

Figure 8a illustrates the progression of the correspondence function learned by the *rotation* generator, which consistently mapped the source task to the target task correctly. In contrast, Figure 8b indicates the slower, but ultimately reasonable, progress made by the *resnet* generator, further emphasizing the benefits of the *rotation* generator in establishing task similarity.

Lastly, Figure 9 compares the performance of the agent in the rotated Pong task under different conditions:

1) When learning from scratch, performance improves slowly over time.
2) When using the correspondence function learned by the *resnet* generator, performance improves much faster at the start.
3) When using the *rotation* generator, the agent achieves immediate reuse of prior knowledge and performs at a high level from the beginning.
4) When fine-tuning the Q-function from the source task without considering correspondence, performance was worse than learning from scratch, likely due to overfitting to the source task.

This comparison highlights the advantages of the proposed method, which enables seamless knowledge transfer, significantly reduces training time, and improves initial performance. In contrast, fine-tuning the Q-function (used as one of the baselines) without proper alignment between tasks leads to poor results, underscoring the importance of task-specific correspondence functions.

V. CONCLUSION AND DISCUSSION

This paper presented a novel method for efficient one-to-one knowledge transfer between reinforcement learning tasks. Our approach modifies CycleGAN specifically for reinforcement learning by incorporating a new loss function that includes the Q-function and environment model from the source task. Through experiments on the 2-D Atari game Pong, we demonstrated that our method outperforms baseline models such as GAN and CycleGAN, providing faster learning and better performance, particularly in scenarios where task environments differ.

One of the key findings of this work is the importance of the network architecture when learning the correspondence function. While both the rotation-based and convolutional generators achieved reasonable results, the rotation-based generator yielded superior performance. This suggests that convolutional layers, commonly used in image-based tasks, may not be optimal for reinforcement learning transfer learning tasks. Future research should explore other architectures, such as transformers, [40], which may further improve generalisation.

In comparison to other knowledge transfer methods, our approach has the advantage of being applicable to a variety of domains without the need for paired data, allowing it to handle diverse RL tasks with varying state formats. However, we acknowledge some limitations. The current method struggles with tasks that have low similarity, and we have not yet explored transferring knowledge from multiple source tasks or automatically selecting the most relevant source task.

Future Directions:

1) **Expanding the validation**: testing the proposed method on a broader range of RL tasks to assess its generalization ability and robustness.
2) **Knowledge transfer in low-similarity tasks**: investigating how to transfer knowledge between tasks with low similarity.
3) **Identifying relevant knowledge**: exploring methods to identify and transfer relevant knowledge from multiple source tasks.
4) **Source task selection**: developing strategies for selecting the most relevant source tasks for transfer.
5) **Alternative network architectures**: researching alternative network architectures, like transformers, to enhance correspondence learning.

In conclusion, our approach represents a significant advancement toward practical and flexible knowledge transfer in reinforcement learning. However, several challenges remain that future work must address to enhance the robustness and adaptability of such systems.

Method implementation: The method implementation in Python is available at *https://github.com/marko-ruman/RL-Correspondence-Learner*

Data availability statement: The datasets generated and/or analysed during the current study are available from the corresponding author on reasonable request.

Ethical approval: This article does not contain any studies with human participants performed by any of the authors.

VI. DQN ALGORITHM

Algorithm 1 summarises the DQN algorithm used in the text (see Section II-C for details). θ denotes parameters of the source network and θ^T are parameters of the target network. Both networks have the same architecture.

Algorithm 1 DQN

Input: initial parameters θ of Q-function $Q(s, a, \theta)$, learning rate $\alpha \in (0, 1)$, discount factor $\gamma \in (0, 1)$, exploration rate $\epsilon \in (0, 1)$, size of the experience memory n_M, size of the learning mini-batch n_B, number of steps for target network synchronization n_U

Initialize experience memory size n_M
Set parameters of the target network $\theta^T = \theta$
for $t = 1, 2, \ldots,$ till convergence **do**
  With exploration ϵ perform random action a_t otherwise select $a_t = \operatorname{argmax}_{a \in a} Q(s, a \mid \theta)$
  Get next state s_{t+1} and reward r_t
  If the memory is full, remove the oldest data record
  Store (s_t, a_t, r_t, s_{t+1}) in experience memory **M**
  Sample a random mini-batch of size n_B $(s_j, a_j, r_j, s_{j+1})_{j \in Rand(n_B)} \in M$
  for every j **do**
    if s_{j+1} is a terminal state **then**
      $\text{target}_j = r_j$
    else
      $\text{target}_j = r_j + \gamma \max_{a' \in a} Q(s_{j+1}, a' \mid \theta^T)$
    end if
  end for
  Perform a gradient descent on $\left(\left(\text{target}_j - Q(s_j, a_j \mid \theta)\right)^2\right)_{j \in Rand(n_B)}$ with Huber loss, [41], with respect to parameters θ
  Every n_U steps set $\theta^T = \theta$
end for

Output: Q-function $Q(s, a)$, experience memory **M**

VII. IMPLEMENTATION DETAILS

A. CYCLEGAN GENERATOR ARCHITECTURES

The architectures of generators G_S and G_T in Experiment 1 (Section IV-C) and the resnet generators G_S and G_T in Experiment 2 (Section IV-D) were taken from [25]. The 9 residual blocks version was used. Below, we follow the naming convention used in [25].

Let `c7s1-f` denote a 7×7 Convolution-BatchNorm-ReLU layer with f filters and stride 1. `df` denotes a 3×3 Convolution-BatchNorm-ReLU layer with f filters and stride 2. Reflection padding was used to reduce artefacts. `Rf` denotes a residual block that contains two 3×3 convolutional layers with the same number of filters (f) on both layers. `uf` denotes a 3×3 fractional-strided-Convolution-BatchNorm-ReLU layer with f filters and stride 2.

The network architecture consisted of:

`c7s1-64,d128,d256,R256,R256,R256,R256,R256,R256,R256,R256,u128,u64,c7s1-3`

The *rotation* generator contained just one rotation layer, see [39].

B. DISCRIMINATOR ARCHITECTURES

For discriminator networks D_S and D_T in all the experiments, 70×70 PatchGAN was used, see [38]. Let `Cf` denote a 4×4 Convolution-BatchNorm-LeakyReLU layer with f filters and stride 2. After the last layer, a convolution to produce a 1-dimensional output was used. Leaky ReLUs were used with a slope of 0.2.

The discriminator architecture was:

`C64,C128,C256,C512`.

C. Q-FUNCTION ARCHITECTURE

Q-function had architecture taken from [3]. Let `c-k-s-f` denote a $k \times k$ Convolution-ReLU layer with stride s and f filters and `f-o` is a Fully connected-ReLU layer with o outputs. The Q-function architecture was:

`c-8-1-32,c-4-2-64,c-3-1-64,f-512,f-6`.

D. ENVIRONMENT MODEL ARCHITECTURE

The environment model F had the same architecture as the generators G_S and G_T with one difference: the fifth residual block received one-hot encoded actions as an additional input.

The architecture of the environment model was then as follows:

`c7s1-64,d128,d256,R262,R262,R262,R262,R262,u128,u64,c7s1-3`.

VIII. TRAINING

All the networks are trained from scratch with weights initialized from a Gaussian distribution $N(0, 0.02)$.

The environment model, F, was trained with Adam optimizer, [42], with the learning rate of 0.001, batch size of 16 and it was trained for 50 epochs.

For the training of the Q-function, RMSprop optimiser, [43], was used. The learning rate was 0.0001, and the batch size was 32. The other parameters of Q-learning were identical to those in [3].

Generators G_S and G_T and discriminators D_S and D_T were jointly trained using Adam optimizer with an initial learning rate of 0.0002 which was linearly decayed to zero. The training took four epochs.

REFERENCES

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[2] A. R. Mahmood, D. Korenkevych, and G. Vasan, "Benchmarking reinforcement learning algorithms on real-world robots," in *Proc. Conf. Robot Learn.*, 2018, pp. 561–591.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[4] Y. Hu, S. Sun, X. Xu, and J. Zhao, "Attentive multi-view reinforcement learning," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 11, pp. 2461–2474, Nov. 2020.
[5] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.
[6] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 2555–2565.
[7] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Jan. 2009.
[8] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13344–13362, Nov. 2023.
[9] H. B. Ammar, E. Eaton, J. M. Luna, and P. Ruvolo, "Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 1–7.
[10] N. V. Varghese and Q. H. Mahmoud, "A hybrid multi-task learning approach for optimizing deep reinforcement learning agents," *IEEE Access*, vol. 9, pp. 44681–44703, 2021.
[11] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12619–12629.
[12] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai, "RetinaGAN: An object-aware approach to sim-to-real transfer," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 10920–10926.
[13] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "RL-CycleGAN: Reinforcement learning aware simulation-to-real," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11154–11163.
[14] W. Zhu, X. Guo, D. Owaki, K. Kutsuzawa, and M. Hayashibe, "A survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 7, pp. 3444–3459, Jul. 2023.
[15] M. Ranaweera and Q. H. Mahmoud, "Bridging the reality gap between virtual and physical environments through reinforcement learning," *IEEE Access*, vol. 11, pp. 19914–19927, 2023.
[16] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is enough," *Artif. Intell.*, vol. 299, Oct. 2021, Art. no. 103535.
[17] J. Clune, "AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence," 2019, *arXiv:1905.10985*.
[18] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.
[19] X. Qu, Z. Sun, Y.-S. Ong, A. Gupta, and P. Wei, "Minimalistic attacks: How little it takes to fool deep reinforcement learning policies," *IEEE Trans. Cognit. Develop. Syst.*, vol. 13, no. 4, pp. 806–817, Dec. 2021.
[20] S. Gamrian and Y. Goldberg, "Transfer learning for related reinforcement learning tasks via image-to-image translation," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2063–2072.
[21] B. Chen, A. Sax, G. Lewis, I. Armeni, S. Savarese, A. Zamir, J. Malik, and L. Pinto, "Robust policies via mid-level visual representations: An experimental study in manipulation and navigation," 2020, *arXiv:2011.06698*.
[22] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," 2017, *arXiv:1703.02949*.
[23] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.
[24] X. Gao, J. Si, and H. Huang, "Reinforcement learning control with knowledge shaping," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 3, pp. 3156–3167, Mar. 2024.
[25] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2242–2251.
[26] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–9.
[27] Q. Zhang, T. Xiao, A. A. Efros, L. Pinto, and X. Wang, "Learning cross-domain correspondence for control with dynamics cycle-consistency," 2020, *arXiv:2012.09811*.
[28] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh, "Recycle-GAN: Unsupervised video retargeting," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 119–135.
[29] M. L. Puterman, "Markov decision processes," in *Handbooks in Operations Research and Management Science*, vol. 2. New York, NY, USA: Wiley, 1990, pp. 331–434.
[30] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, May 1992.
[31] B. C. Csáji, "Approximation with artificial neural networks," *Fac. Sci., Etvs Lornd Univ., Hung.*, vol. 24, no. 48, p. 7, 2001.
[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.
[35] A. N. Meltzoff, "Understanding the intentions of others: Re-enactment of intended acts by 18-month-old children," *Develop. Psychol.*, vol. 31, no. 5, pp. 838–850, 1995.
[36] 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, Jun. 2013.
[37] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. 14th Eur. Conf. Comput. Vis. (ECCV)*. Amsterdam, The Netherlands: Springer, Oct. 2016, pp. 694–711.
[38] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5967–5976.
[39] M. Jaderberg, K. Simonyan, and A. Zisserman, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.
[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.
[41] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in Statistics*. Berlin, Germany: Springer, 1992, pp. 492–518.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[43] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*.

MARKO RUMAN received the Ing. (equivalent to M.Sc.) degree in mathematical engineering from Czech Technical University, Prague, Czech Republic, in 2018, where he is currently pursuing the Ph.D. degree in mathematical engineering.

He is also a Research Assistant with the Department of Adaptive Systems, Institute of Information Theory and Automation, Czech Academy of Sciences. His main research interests include knowledge transfer and reinforcement learning.

TATIANA V. GUY (Senior Member, IEEE) received the Dipl.-Eng. degree in control and automation from Kiev Polytechnic Institute, and the Ph.D. degree in cybernetics from Czech Technical University, Prague.

She is currently with the Institute of Information Theory and Automation, Prague. Since 2013, she has been the Head of the Adaptive Systems Department. She has also an appointment as an Associate Professor with Czech University of Life Sciences. Her current research interests include distributed decision making, nature-inspired cooperation, and transfer learning.