# Transfer Learning for Deep Reinforcement Learning

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Deep reinforcement learning (RL) has shown the potential to achieve superhuman performance in solving complex decision tasks. Although, unlike humans, it fails to generalise and reuse previously acquired knowledge effectively, which is a crucial ability for a truly intelligent agent. The work proposes an RL-specific modification of CycleGAN, which ensures one-to-one knowledge transfer between different RL tasks. We evaluate the approach on the 2-D Atari game Pong and compare it against two baselines: using GAN and CycleGAN methods. The results demonstrate that our method consistently outperforms the state-of-the-art methods.

## 1 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 RL advantages with the power to handle high-dimensional data, recently brought many advances. For instance, model-free methods show significant results in 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], or PlaNET, [6] made significant progress. However, in many real-world tasks, RL remains unsuitable as the errors can be extremely costly. One of the promising ways to address this issue is using *transfer learning* (TL), [7] when skills and knowledge collected on similar tasks are applied to the currently solved problem. Besides, learning is essential for developing agents capable of lifelong learning, [8], for simulation-to-real knowledge transfer used in robotics, [9, 10, 11, 12], or for developing the general AI, [13].

Despite many advances made, the use of transfer learning in RL and especially TL in deep RL, is limited. For example, 1-pixel perturbations of state observations can lead to useless policies, [14]. The RL methods often fail to reuse previously acquired knowledge even in similar tasks when the original image is rotated, or some colours are changed. It has also been shown that learning from scratch can be more efficient than fine-tuning a previously obtained model, [15]. That significantly contrasts with the human ability to generalise and reuse previously acquired knowledge.

### Main contributions of the paper

- It introduces a *method for knowledge transfer between two different RL tasks* based on RL-specific modification of CycleGAN. The method is highly applicable in practice. The *method does not rely on paired data* and is independent of the nature of the involved RL tasks. It ensures that the approach can be easily applied to various domains.
- The work establishes a *correspondence function that reveals similarity of the source and target RL tasks*. The proposed formulation suggests learning the correspondence function by minimising the discriminative loss function.

- The work proposes a *new four-component loss function* which reflects different types of losses. The proposed modification accounts for the actual policy used and takes into account the existing dynamic relationships of the involved RL tasks.
- We demonstrate how adding two new components *generalises GAN and CycleGAN methods*, i.e. the latter are special cases of the proposed approach.
- We achieve results indicating a complete reuse of previously acquired knowledge when transferring between the original Pong and rotated Pong.
- We show that the proposed approach copes with the tasks for which standard approaches (GAN, CycleGAN) fail and learning from scratch remains to be the most efficient method.

Experiments with 2-D Atari game Pong, [16], demonstrated that the proposed method notably speeds up the learning and increases the average reward.

The paper layout is as follows. Section 2 recalls the necessary background and formulates the TL problem. Section 3.4 constructs the correspondence function and proposes a novel method of its learning. Section 4 describes the experimental evaluation of the proposed approach and compares it with baseline methods. Section 5 provides concluding remarks and outlines future research directions.

**Related works**

Survey [17] systematically analyses recent advances in transfer learning for deep RL. The research category to which our approach belongs utilises mapping functions between the source and target task to ensure knowledge transfer. Among them, a line of research that learns common features of RL tasks that can be transferred. It was shown, [18], that the policies learnt on so-called mid-level features can generalise better than those learnt directly on image observations. Work [19] 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 [20] achieved success in tasks differing in reward function by maintaining successor features and decoupling environment dynamic and reward function. Approach [21] 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[1], [22], and UNIT[2], [23]. Approach [15] 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 were present. Works [11] and [10] improved the approach proposed in [15] by introducing learnt $Q$-function or object detection into the learning of the task correspondence. One of the recent approaches, [24], includes an environment model in the learning of the task correspondence. This approach is strongly inspired by the video-to-video translation model, [25].

## 2 Background and notation

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

### 2.1 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).

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.

[1]Cycle generative adversarial network

[2]Unsupervised Image-to-Image Translation Networks

### 2.2 Reinforcement learning

Reinforcement learning (RL) considers an *agent* purposefully interacting with an *environment* by selecting actions. RL agent models its environment as a *Markov decision process* (MDP), [26] 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 $s_t \in \mathbf{S}$ and takes action $a_t \in \mathbf{A}$. Executing action $a_t$ at state $s_t$: i) causes a transition 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 $R(s_{t+1}, a_t, s_t)$. The agent's *goal* is to learn policy $\pi^* : \mathbf{S} \mapsto \mathbf{A}$ that maximises the accumulated reward.

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, [27].

Deep $Q$-networks (DQN), [28], use a standard off-policy $Q$-learning, [29], and DNN to estimate the $Q$-function, which then gives the maximizing policy $\pi^*$.

## 3 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[3]. Generally finding common patterns between different tasks and effectively transferring the concepts learned on one task to another is an essential characteristic of high-level intelligence.

In this section, we formalise a problem of transfer learning between two RL tasks, empirically introduce a correspondence function reflecting the similarity of two RL tasks and propose 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.

### 3.1 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 2.2. Transition functions of the tasks as well as theirs reward functions may be different.

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

$$Q_S(\mathcal{C}(.,.)) : \mathbf{S}_T \times \mathbf{A}_T \mapsto \mathrm{R} \tag{1}$$

gives better performance[4] 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[5]. Thus, we need to learn a mapping indicating corresponding states, i. e. the

[3]Developmental psychologists have shown that as early as 18 months old, children can infer intentions and imitate the behaviour of adults, [30]. The imitation is complex as children must infer a match between their observations and internal representations, effectively linking the two diverse domains.

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

[5]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

correspondence function for states. The searched correspondence function $\mathcal{C}$ is then obtained as follows:

$$\mathcal{C}(s_T, a_T) = (G_T(s_T), I(a_T)), \qquad \forall (s_T, a_T) \in \mathbf{S}_T \times \mathbf{A}_T, \tag{2}$$

where $G_T$ is the generator from (3) 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.

### 3.2 Learning of correspondence function

The proposed learning is inspired by CycleGAN, see Section 3.3, 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 [15]. 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 making the method entirely relevant to RL.

This work proposes adding two new components to the CycleGAN losses, (4), (5):

- $Q$-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 briefly summarize CycleGAN and explain the reasons for introducing the new components and their forms.

### 3.3 CycleGAN

Cycle-consistent Generative Adversarial Network (CycleGAN), [22], is based on GAN[6], [31], 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*[7]

$$G_S: \ \mathbf{S}_S \to \mathbf{S}_T \quad \text{and} \quad G_T: \ \mathbf{S}_T \to \mathbf{S}_S. \tag{3}$$

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.

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

$$\begin{aligned}\mathcal{L}_{GAN} = & E_{s_S}\left[\log D_S(s_S)\right] + E_{s_T}\left[\log\left(1 - D_S\left(G_S\left(s_T\right)\right)\right)\right] \\ & + E_{s_T}\left[\log D_T(s_T)\right] + E_{s_S}\left[\log\left(1 - D_T\left(G_T\left(s_S\right)\right)\right)\right]\end{aligned} \tag{4}$$

The adversarial training encourages mappings $G_S$ and $G_T$ (3) to produce outputs indistinguishable from the real ones, i. e. respective sets $\mathbf{S}_S$ and $\mathbf{S}_T$.

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

$$\mathcal{L}_{Cyc} = E_{s_S}\left[\|G_T\left(G_S\left(s_S\right)\right) - s_S\|\right] + E_{s_T}\left[\|G_S\left(G_T\left(s_T\right)\right) - s_T\|\right]. \tag{5}$$

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 $s_T \in \mathbf{S}_T$.

[6]Generative adversarial network

[7]that translate data between source and target domains

***Q−loss***

The available $Q$-function, $Q_S$, should be incorporated in the learning of correspondence function $\mathcal{C}$ as it determines the optimal policy. We introduce loss $\mathcal{L}_Q$ in the following form:

$$\mathcal{L}_Q = E_{s_S} \left[ \| Q_S \left( G_T \left( G_S \left( s_S \right) \right) \right) - Q_S \left( s_S \right) \| \right] \tag{6}$$

The loss (6) will make the learned correspondence more suitable for transferring knowledge between tasks $S$ and $T$ because the learned correspondence function, $\mathcal{C}$, will retain the parts of the states that are the most important for choosing the optimal action.

***Model loss***

So far, all considered losses (4) - (6) are associated with state values. However, every RL task is a dynamic one, and the time sequence of states is essential. Consider states of the target and the source tasks at times $t$ and $t+1$. If generator $G_T$ ensures mapping $s_{Tt}$ on $s_{St+1}$ and generator $G_S$ maps $s_{St+1}$ back to $s_{Tt}$, then losses $\mathcal{L}_{GAN}, \mathcal{L}_{Cyc}$ and $\mathcal{L}_Q$ (4) - (6) are minimal. However, it would not help to solve the target RL task.

To ensure that the correspondence function, $\mathcal{C}$, grasps all essential dynamic relationships of the source and target task, the overall loss minimised must consider the learnt environment model, $F$[8] of the source task:

$$\mathcal{L}_M = E_{s_{Tt}, a_{Tt}, s_{Tt+1}} \left[ \| F \left( G_T \left( s_{Tt} \right), a_{Tt} \right) - G_T \left( s_{Tt+1} \right) \| \right] \tag{7}$$

***Total loss***

The proposed total loss comprises all the components (4), (5), (6) and (7) and, thus, has the following form:

$$\mathcal{L} = \mathcal{L}_{GAN} + \lambda_{Cyc} \mathcal{L}_{Cyc} + \lambda_Q \mathcal{L}_Q + \lambda_M \mathcal{L}_M, \tag{8}$$

where $\lambda_{Cyc}$, $\lambda_Q$ and $\lambda_M$ are *loss parameters* that define relative influence (weight) of the respective components.

The proposed approach, which minimises 4-component loss (8), generalises GAN, [31], and CycleGAN, [22], methods often used for transfer learning. It is easy to see that GAN and CycleGAN can be obtained by setting some of parameters $\lambda_Q$, $\lambda_M$, $\lambda_{Cyc}$ in (8) to zeros as follows:

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

### 3.4 Transfer learning: Algorithm

The main steps of the proposed algorithm:

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

***Step 2*** The agent applies a random decision rule to task $T$, collects experience memory $\mathbf{M}_T$.

***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 $\mathcal{C}$.

***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:

$$Q_T(s_T, a_T) = Q_S(\mathcal{C}(s_T, a_T)), \ \forall (s_T, a_T) \in (\mathbf{S}_T \times \mathbf{A}_T). \tag{9}$$

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

[8] Environment model $F$: $\mathbf{S} \times \mathbf{A} \mapsto \mathbf{S}$ is a mapping taking current state $s_t$ and action $a_t$ and giving the next state $s_{t+1}$. It is learned using the experience memory, $\mathbf{M}_S$

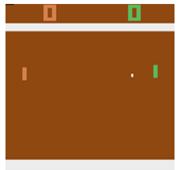


Figure 1: **Standard Pong**, [16]

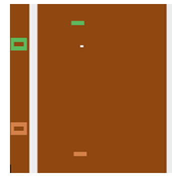


Figure 2: **Pong rotated by 90 degrees**, [16]

# 4 Experimental part

To test the efficiency of the proposed approach, two experiments on the Atari game Pong, [16], 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.

## 4.1 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 Fig. 1). 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 Fig. 2). 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 3.4,
4) For every 1000 learning steps, the agent:
   - suspends learning of correspondence function $\mathcal{C}$,
   - uses learnt $\mathcal{C}$ and the $Q$-function transformed from the *source task*, see (9), 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[9] 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.

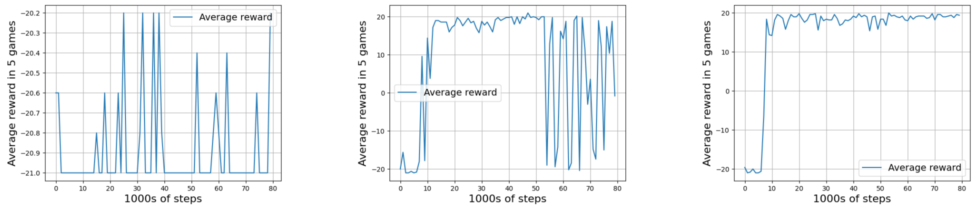
**Baselines:** The results are compared with two baselines - using GAN and CycleGAN methods, [31], [22], which have been recently used for knowledge transfer in similar settings, [15].

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

## 4.2 Experiment 1

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

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




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

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

c) Loss (8) with $\lambda_{Cyc} = 1$,
$\lambda_Q = 1$, $\lambda_M = 1$

Figure 3: **Experiment 1**: Average accumulated reward per game when playing five games with the transformed $Q$-function (9) every 1000 learning steps. The performance is shown for different values of loss parameters $\lambda_{Cyc}$, $\lambda_Q$ and $\lambda_M$. Fig. 3a and 3b show the **baselines** using *GAN* and *CycleGAN* methods.

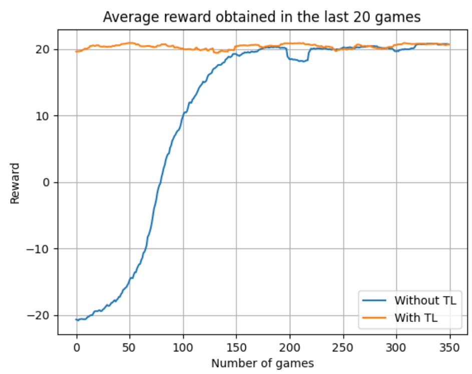



Figure 4: Moving average of reward per game computed from the last 20 games depending on the number of *Pong* games played. The blue line denotes learning from scratch, i. e. without TL, the orange line denotes the case with TL. The $Q$-function $Q_T$ is continuously learned during the game in both cases.

$G_S$ and $G_T$ generators (see Section 3.2) were constructed as neural networks with convolutional layers. Their specific architecture was taken from [32]. The discriminators $D_S$ and $D_T$ were also constructed as neural networks with convolutional layers with the architecture as in [33].

The transfer learning with the loss (8) was tested for all the combinations of the parameters: $\lambda_{Cyc} \in \{0, 1, 10\}$, $\lambda_Q \in \{0, 1\}$ and $\lambda_M \in \{0, 1, 10\}$.

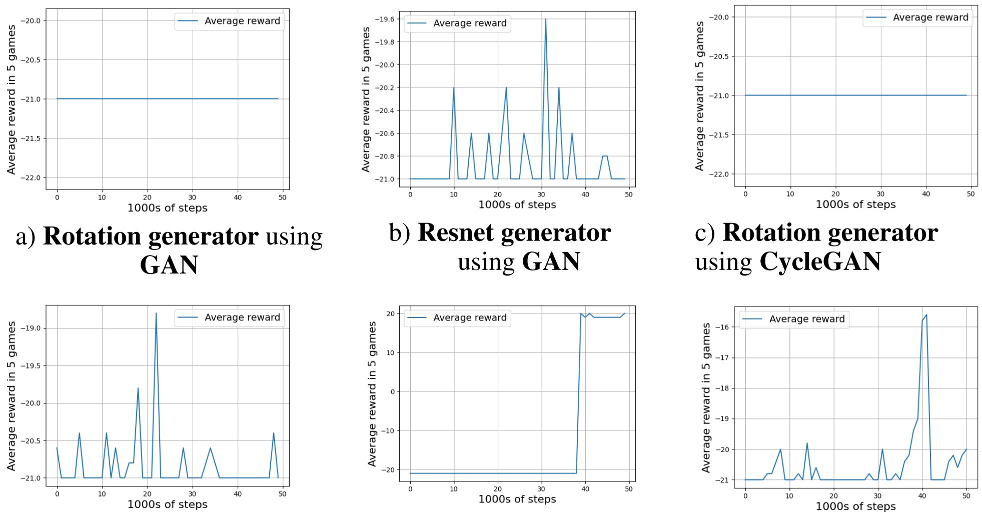
The results are presented in Fig. 3 and Fig. 4. The main findings are:

- The best results are obtained for the proposed loss function, (8), that contains the proposed components $\mathcal{L}_Q$ and $\mathcal{L}_M$, i. e. $\lambda_{Cyc} = \lambda_Q = \lambda_M = 1$, Fig. 3e.
- *GAN baseline*, Fig. 3a, does not produce good results. Performance of *CycleGAN baseline*, Fig. 3b, soon became unstable though it provides good rewards at the beginning.
- The agent successfully learned the correspondence function and completely reused previously acquired knowledge, Fig. 4.

### 4.3 Experiment 2

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

Generators $G_S$ and $G_T$, (see (3) and Section 3.2) are constructed as neural networks with two different architectures (for both of them). The architecture of the first one, referred to here as the




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 5: Average accumulated reward in five games when playing Rotated Pong with the transformed $Q$-function (9) every 1000 learning steps. 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).

**resnet generator**, was taken from [32] and then followed by a rotation layer, see [34]. The second type, referred to as the **rotation generator**, was composed of the mentioned rotation layer only.

The proposed approach was tested for different values of loss parameters $\lambda_{Cyc}$, $\lambda_Q$ and $\lambda_M$, (8). Fig. 5 - Fig. 7 present the best-reached performance compared with the performance of the baseline methods.

The main findings are the following:

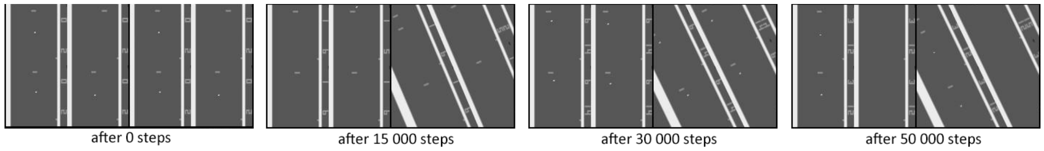
- The GAN and CycleGAN baselines did not produce a correspondence function suitable for knowledge transfer.
- The rotation generator yields the perfect correspondence function. Learning the correspondence function with the **resnet generator** provided significantly better results than learning the $Q$-function from scratch.
- Fine-tuning the $Q$-function from the source task gives worse performance on the target task than learning a new $Q$-function from scratch.

## 5 Conclusion

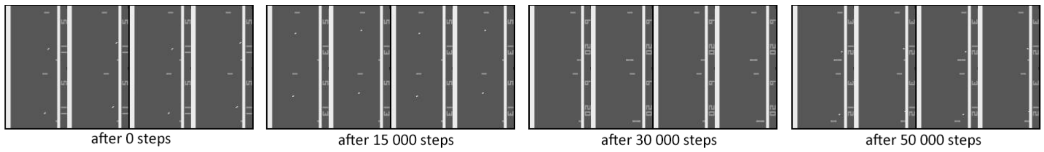
We propose a method for knowledge transfer between two different reinforcement learning tasks. Our approach establishes the correspondence function that reveals the similarity between the *source* and *target task*. The neural network approximates the correspondence function and learns it from unpaired data using dynamic cycle consistency. To ensure that the essential dynamic relationships between the involved RL tasks are exploited, we have introduced a four-component loss (8) with two novel components: model loss and $Q$-loss.

We show the efficacy of our approach on simulated experiments involving the 2-D Atari game Pong and compare it against two baselines using GAN and CycleGAN methods.

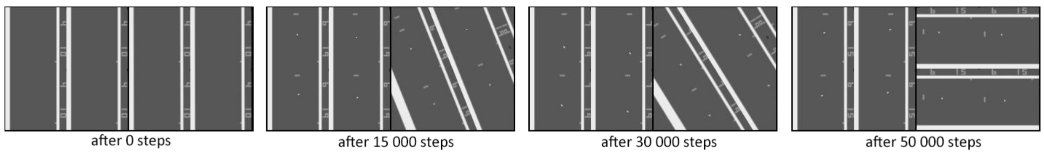
The results show that the proposed approach outperforms baseline methods. The introduced correspondence function respects $Q$-function and environment model of the source task and establishes them into learning. This allows the agent to gradually build, adapt, and use a set of skills while



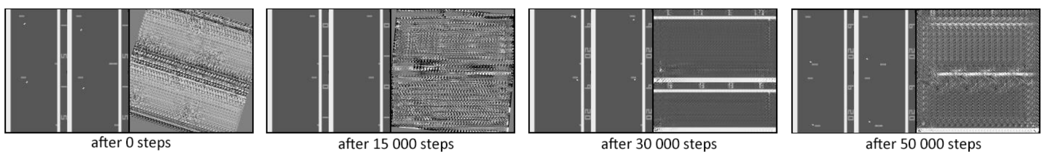

a) Rotation generator using **GAN**



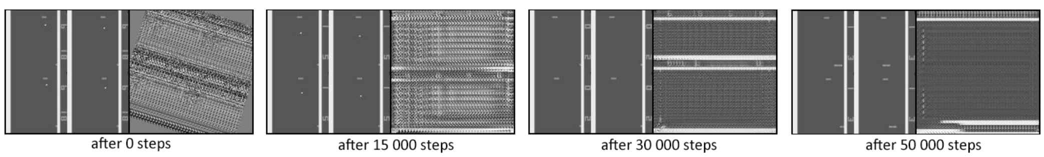

b) Rotation generator using **CycleGAN**



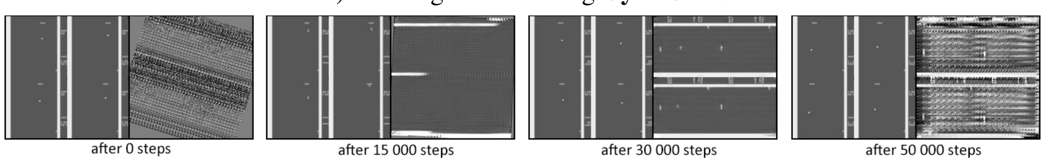

c) Rotation generator with $\lambda_{Cyc} = 0$, $\lambda_Q = 0$, $\lambda_M = 10$




d) Resnet generator using **GAN**




e) Resnet generator using **CycleGAN**




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

Figure 6: **Experiment 2**: Screenshots of the game depicting progress in learning the correspondence function $\mathcal{C}$ (2) after 0, 15000, 30000 and 50000 steps. The results are shown for the rotation and the resnet generators with the best settings of parameters $\lambda_{Cyc}$, $\lambda_Q$ and $\lambda_M$ (8) 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 $\mathcal{C}$.

interacting with the dynamically changing environment, which is generally different from the source task.

The most significant advantage of the proposed method is its practical applicability. The solution does not rely on paired data and is independent of the nature of the involved RL tasks. It ensures that the approach can be easily applied to various domains.

The foreseen research should focus on the open problems: i) how to perform transfer learning between tasks having low similarity, ii) how to identify and transfer relevant knowledge from several source tasks, iii) how to choose the only relevant source tasks similarly to [35], iv) what is a better network architecture for the correspondence function learning.

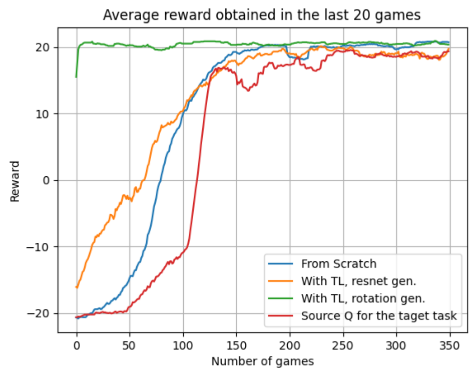



Figure 7: Moving average of reward per game computed from the last 20 games depending on the number of played games for the game rotated Pong for four different agents - an agent learning the game from scratch (blue line), an agent using the correspondence function learned with the resnet generator (orange 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.

**Method implementation**: The method implementation in Python is available at *https://github.com/*** (anonymized)

# References

[1] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[2] A Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking reinforcement learning algorithms on real-world robots. In *Conference on Robot Learning*, pages 561–591. PMLR, 2018.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[4] Yueyue Hu, Shiliang Sun, Xin Xu, and Jing Zhao. Attentive multi-view reinforcement learning. *International Journal of Machine Learning and Cybernetics*, 11(11):2461–2474, 2020.

[5] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[6] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[7] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[8] Haitham Bou Ammar, Eric Eaton, José Marcio Luna, and Paul Ruvolo. Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In *Twenty-fourth International Joint Conference on Artificial Intelligence*, 2015.

[9] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In

*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[10] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. Retinagan: An object-aware approach to sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10920–10926. IEEE, 2021.

[11] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.

[12] Wei Zhu, Xian Guo, Dai Owaki, Kyo Kutsuzawa, and Mitsuhiro Hayashibe. A survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[13] Jeff Clune. AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*, 2019.

[14] Xinghua Qu, Zhu Sun, Yew-Soon Ong, Abhishek Gupta, and Pengfei Wei. Minimalistic attacks: How little it takes to fool deep reinforcement learning policies. *IEEE Transactions on Cognitive and Developmental Systems*, 13(4):806–817, 2020.

[15] Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. In *International Conference on Machine Learning*, pages 2063–2072. PMLR, 2019.

[16] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[17] Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13344–13362, 2023.

[18] Bryan Chen, Alexander Sax, Gene Lewis, Iro Armeni, Silvio Savarese, Amir Zamir, Jitendra Malik, and Lerrel Pinto. Robust policies via mid-level visual representations: An experimental study in manipulation and navigation. *arXiv preprint arXiv:2011.06698*, 2020.

[19] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

[20] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[21] Xiang Gao, Jennie Si, and He Huang. Reinforcement learning control with knowledge shaping. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2023.

[22] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.

[23] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *Advances in Neural Information Processing Systems*, 30, 2017.

[24] Qiang Zhang, Tete Xiao, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Learning cross-domain correspondence for control with dynamics cycle-consistency. *arXiv preprint arXiv:2012.09811*, 2020.

[25] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–135, 2018.

[26] Martin L Puterman. Markov decision processes. *Handbooks in operations Research and Management Science*, 2:331–434, 1990.

[27] Balázs Csanád Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7, 2001.

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[29] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[30] Andrew N Meltzoff. Understanding the intentions of others: re-enactment of intended acts by 18-month-old children. *Developmental Psychology*, 31(5):838, 1995.

[31] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[32] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[33] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[34] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in Neural Information Processing Systems*, 28, 2015.

[35] Marzieh Davoodabadi Farahani and Nasser Mozayani. Evaluating skills in hierarchical reinforcement learning. *International Journal of Machine Learning and Cybernetics*, 11(10):2407–2420, 2020.