# Transfer Learning for Dynamical Systems Models via Autoencoders and GANs

Angelo Damiani
*Gran Sasso Science Institute*
L'Aquila, Italy
angelo.damiani@gssi.it

Gustavo Viera Lopez
*Gran Sasso Science Institute*
L'Aquila, Italy
gustavo.vieralopez@gssi.it

Giorgio Manganini
*Fraunhofer Research Italia*
Bolzano, Italy
giorgio.manganini@fraunhofer.it

Alberto Maria Metelli
*Politecnico di Milano*
Milano, Italy
albertomaria.metelli@polimi.it

Marcello Restelli
*Politecnico di Milano*
Milano, Italy
marcello.restelli@polimi.it

***Abstract*—Transfer learning has seen significant progress in the domains of supervised learning and reinforcement learning, yet there remains a noticeable gap in the area of regression. In supervised learning, various transfer learning methods have been developed to leverage knowledge from one task to improve performance on another, but these approaches are primarily designed for classification tasks. In the realm of reinforcement learning, many techniques focus on policy transfer, and those that do address samples transfer predominantly apply it within homogeneous contexts. This paper presents a novel algorithm that bridges the transfer learning gap between supervised learning and reinforcement learning while specifically addressing the regression problem of dynamical systems' model estimation. Our approach harnesses the feature extraction capabilities of autoencoders and the generative power of Generative Adversarial Networks (GANs) to train a mapping that facilitates the seamless transformation of samples between dynamical systems. This approach represents a significant advancement in the field of transfer learning, as it offers a versatile and effective solution for transferring regression models across heterogeneous domains. Our experimental results demonstrate the algorithm's efficacy and its potential to improve model generalization and adaptability in diverse scenarios with varying data distributions and dynamics.**

***Index Terms*—Transfer Learning, Reinforcement Learning, Markov Decision Process, Autoencoders, Model Transfer, Instance Transfer**

## I. Introduction

Transfer learning (TL) has emerged as a transformative paradigm in the realm of machine learning, promising of leveraging knowledge from one domain to enhance performance in another. In the context of supervised learning, TL has garnered considerable attention and success, particularly for classification tasks [16], [28]. Numerous pioneering techniques have been developed to facilitate knowledge transfer for classification, enabling models to achieve impressive results in diverse domains, such as recommender systems [32], medical decision-making and analysis [29], [7], and text classification [14]. Despite these successes, almost no attention has been directed towards leveraging TL techniques for regression tasks [28], and the potential for extending TL to regression problems remains largely untapped and dealt with just in a few works [18], [4].

TL research has also been increasingly applied to the Reinforcement Learning (RL) [19] paradigm, exploring how RL algorithms can take advantage of knowledge transfer [9], [34] in contexts like robotics [33] or IoT [15]. While traditional RL allows for learning solutions to any task independently, the amount of samples necessary to acquire a near-optimal solution can be impractical in real-world applications unless prior knowledge is available. In these situations, TL may play a decisive role in mitigating the possible lack of data. In the RL paradigm, the fundamental concept behind transfer learning is that the agent does not have to begin learning a desired task from scratch. Instead, the agent can first be trained on one or more source tasks, i.e. Markov Decision Processes (MDPs) [17], and subsequently transfer the acquired knowledge to facilitate the solution of the target task. This knowledge can take various forms, including samples [11], [10], policies [3], or value functions [22]. By leveraging the knowledge obtained from previous tasks, an agent can tackle the new task more effectively.

Depending on the MDP's state and action spaces (i.e., the domain) of both source and target tasks, a TL approach can be classified to be either *inter-domain* or *cross-domain*. The former aims to transfer knowledge between tasks sharing the same domain but described by different dynamics or goals [11], [27], [26]. In the latter, besides different dynamics and goals, the tasks have different domains, and the state-action variables can vary in shape and range. The transfer approaches, in this case, focus on defining an inter-task mapping function between the source state-action variables and the target ones to achieve an effective transfer of knowledge [25], [24], [20], [6], [31], [30]. Assuming to know a suitable mapping function was a *de-facto* standard for the early stages of cross-domain Transfer Learning in RL [25], [24], [20]. There were some exceptions where the inter-task mapping was either searched iterating over all the 1-to-1 possible state and actions mappings [21], making the method unfeasible for those environments with large domains, or empirically learned using tasks similarities [23].

During the last decade, Transfer Learning research in RL has focused on learning the inter-task mapping in an automatic manner to be applied in a policy transfer setting. This mapping is then utilized either to convert a policy from a source task into one for the target task [31], or to guide the RL process in the target task via reward shaping [6] or mixing sources' policies along with the learning one [30]. In these cases, a pre-established policy acquired from the source environment is required, and it is crucial that the source and target tasks share significant similarities. Finally, most approaches require some degree of interaction with the actual environment, which may restrict the possibility of transfer learning in situations where offline implementation is necessary due to performance and/or safety concerns.

A different approach, known as *instance transferring*, aims to transfer samples between tasks instead of transferring policies, thus overcoming the limitation of requiring a pre-established policy for the source task. Once data is mapped from a source to a target domain, it is possible to use them in multiple ways. For instance, transformed samples can be employed in model-free offline RL algorithms, as well as exploited to identify a transition model that is then leveraged in a model-based approach. However, previous works in this area present some major drawbacks like the assumption of having a hand-coded inter-task mapping [20], or source and target tasks relying on the same state and action space [11], [27], [26]. In [11] the authors presented a multi-source instance transferring algorithm where source samples were selected according to a similarity measure between tasks' data distribution. Transferred samples were then employed by an offline model-free algorithm, along with the target samples. In [27] authors proposed a method, via importance weighting, to measure the relevance of source samples that can facilitate the solution of the target task in a value-based setting. Similarly, in [26] the intricate issue of reusing samples in policy search methods was investigated. By drawing on concepts from multiple importance sampling, they proposed robust gradient estimators that successfully satisfy this purpose, as well as a variety of methods to lower the variance. Despite the promising results, aforementioned approaches are defined only between tasks with similar dynamics and, more critically, with the same state and action spaces. An approach that overcame these issues was presented in [1], where the authors trained an inter-task mapping between heterogeneous domains exploiting the sparse coding technique [12]. Transition triplets from source and target tasks were projected into a higher dimensional space, where target triplets were successively paired with the closest source triplets. This allowed the authors to frame the approximation of the mapping function as a supervised learning problem, solved using sparse Gaussian processes, which are then used to transfer source triplets into target ones. The main drawback of this approach is that, after projecting source and target datasets in the high dimensional informative space, finding pairs of closest projected triplets in this new space is very computationally expensive and makes the solution unfeasible in contexts involving a large amount of data.

In this work, we present a novel instance transfer algorithm that combines the feature extraction capabilities of autoencoders [8] with the generative power of GANs [5], so as to:

i primarily focus on the utilization of mapped samples, in conjunction with the original target data, in order to increase the available sample pool. This augmented data budget aims to enhance the capacity for estimating a transition model that effectively captures the dynamics of the system;

ii address the limitation associated with using identical source and target domains;

iii avoid computationally intensive sub-routines;

iv allows transfer from multiple source tasks thereby exploiting knowledge from diverse domains, resulting in a larger and more informative synthetic dataset.

To evaluate the effectiveness of our approach, we trained different transition models by varying the type of data (target data only, synthetic data only and mix of both) and comparing their prediction errors. The numerical findings indicate that the transition model, trained using both synthetic and target data, exhibits superior accuracy compared to the alternative models. As an additional assessment of our approach, we integrated this improved transition model to supply data in the state-of-the-art RL algorithm Deep Deterministic Policy Gradient (DDPG) [13]. This integration proves particularly valuable in RL scenarios where the original target data alone is insufficient to construct a representative transition model, showcasing the practical applicability and advantages of our method.

## II. Problem Definition

Transfer Learning uses the knowledge gained from multiple tasks to enhance the learning performance of new ones. RL tasks are almost invariably modelled as Markov Decision

Processes (MDPs) [17], which can be described by a tuple $(\mathcal{S}, \mathcal{A}, P, r)$ where $\mathcal{S}$ and $\mathcal{A}$ are sets of states and actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability function, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function measuring the immediate performance of the agent. In this context, a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, defined as a probability distribution over state-action pairs, describes the behavior of the agent in the environment. The goal of an RL agent is to improve its policy to maximize the expected cumulative future rewards.

In our work, we consider the transfer problem between a source task $\mathcal{M}_S = (\mathcal{S}_S, \mathcal{A}_S, P_S, r_S)$ and a target task $\mathcal{M}_T = (\mathcal{S}_T, \mathcal{A}_T, P_T, r_T)$, and compactly refer to their source and target domains as $\mathcal{Z}_S \triangleq \mathcal{S}_S \times \mathcal{A}_S \times \mathcal{S}_S$ and $\mathcal{Z}_T \triangleq \mathcal{S}_T \times \mathcal{A}_T \times \mathcal{S}_T$, respectively. We do not assume any similarity or relationship between source and target domains, in terms of shape and range.

The knowledge transferred in TL approaches may assume several forms. Here we focus on the instance transferring problem, where triplets $z = (s, a, s')$ represent the elements being transferred among the tasks. Specifically, starting from a pair of datasets $\mathcal{D}^S = \{z_S^i \in \mathcal{Z}_S\}^{i=1 \dots n_S}$ and $\mathcal{D}^T = \{z_T^i \in \mathcal{Z}_T\}^{i=1 \dots n_T}$, respectively gathered from source and target tasks, we learn the inter-task mapping function $\mathrm{M} : \mathcal{Z}_S \to \mathcal{Z}_T$ that describes the relationship between tasks. This mapping allows transforming the triplets from the source to the target domain, and in particular to generate, out of the source dataset $\mathcal{D}^S$, a synthetic dataset $\hat{\mathcal{D}}^T = \{\hat{z}_T^i \in \mathcal{Z}_T\}^{i=1 \dots n_S}$ of triplets corresponding to the application of the mapping function $\mathrm{M}$ to the triplets in $\mathcal{D}^S$, i.e., $\hat{z}_T^i = \mathrm{M}(z_S^i)$. This synthetic dataset can be then employed to enhance the learning in the target task w.r.t. the case where only $\mathcal{D}^T$ is available.

## III. Transfer of Instances via Generative Adversarial Autoencoding Networks

We aim to learn an inter-task mapping function $\mathrm{M}$ to be able to transform source triplets $z_S$ to target ones $\hat{z}_T$. For this purpose, we assume to have a very large collection of triplets from the source domain and only a few from the target domain, respectively $\mathcal{D}^S$ and $\mathcal{D}^T$.

We begin by introducing an autoencoder structure [8], where the mapping from the source to the target domain $\mathrm{M} : \mathcal{Z}_S \to \mathcal{Z}_T$ is implemented via the encoder module, while the decoder module implements the inverse mapping $\mathrm{M}^{-1} : \mathcal{Z}_T \to \mathcal{Z}_S$, which helps capture the most salient features of the data. The training of the autoencoder seeks to minimize the L1 loss between the input, i.e., the source triplets $z_S$, and its reconstruction generated as output, $\hat{z}_S = \mathrm{M}^{-1}(\mathrm{M}(z_S))$:

$$J_R(z_S, \hat{z}_S) = \|z_S - \hat{z}_S\|_1 \tag{1}$$

Next, we add two constraints to force the encoder to map the inputs into a region matching the distribution of the target data. Firstly, the encoded triplet $\hat{z}_T = \mathrm{M}(z_S)$ should be consistent with the dynamics seen in the target triplets of $\mathcal{D}^T$. To enforce this requirement, we pre-train an approximate version of the target transition model $\hat{\mathrm{T}}_\mathrm{T} : \mathcal{S}_T \times \mathcal{A}_T \to \mathcal{S}_T$, which takes a state-action pair as input and predicts the next state to approximate the dynamics of the target task, using a supervised regression objective:

$$\min_{\hat{\mathrm{T}}_\mathrm{T}} J_{\text{model}}(\hat{\mathrm{T}}_\mathrm{T}) \triangleq \mathbb{E}_{(s_T, a_T, s'_T) \in \mathcal{D}^T} \left[ \left\| \hat{\mathrm{T}}_\mathrm{T}(s_T, a_T) - s'_T \right\|_2^2 \right] \tag{2}$$

We use this transition model to force the encoder to generate triplets whose $\hat{s}'_T$ component is consistent with the one produced by $\hat{\mathrm{T}}_\mathrm{T}$ when $\hat{s}_T$ and $\hat{a}_T$ are provided. This is done by *minimizing* the following loss function as part of the training process of the autoencoder:

$$J_T(\hat{z}_T; \hat{\mathrm{T}}_\mathrm{T}) \triangleq \left\| \hat{\mathrm{T}}_\mathrm{T}(\hat{s}_T, \hat{a}_T) - \hat{s}'_T \right\|_2^2 \tag{3}$$

Secondly, we train a discriminator $\hat{\mathrm{D}}_\mathrm{T} : \mathcal{Z}_T \to (0, 1]$ designed to classify triplets from the target domain as valid or fake. $\hat{\mathrm{D}}_\mathrm{T}$ must be trained in an adversarial way [5] because its ability to discern should improve together with the encoder's ability to map, which can be obtained by solving the following:

$$\begin{aligned} \max_{\hat{\mathrm{D}}_\mathrm{T}} J_{\text{discr}}(\hat{\mathrm{D}}_\mathrm{T}) = \mathbb{E}_{z_T \sim \mathcal{D}^T} \left[ \log(\hat{\mathrm{D}}_\mathrm{T}(z_T)) \right] + \\ + \mathbb{E}_{z_S \sim \mathcal{D}^S} \left[ \log(1 - \hat{\mathrm{D}}_\mathrm{T}(\mathrm{M}(z_S))) \right] \end{aligned} \tag{4}$$

The problem defined by (4) seeks to improve the recognition of fake triplets by discerning the mapped ones $\mathrm{M}(z_S)$ from the triplets $z_T$ gathered from the target task. The first term of the loss penalizes the recognition of target triplets as fake ones, while the second term penalizes the classification of mapped triplets as target ones. Similarly to what has been done with the approximate transition model $\hat{\mathrm{T}}_\mathrm{T}$, we employ the discriminator $\hat{\mathrm{D}}_\mathrm{T}$ for the definition of an additional loss function to be *minimized* during the train of the autoencoder, i.e.,

$$J_D(\hat{z}_T; \hat{\mathrm{D}}_\mathrm{T}) \triangleq -\log \hat{\mathrm{D}}_\mathrm{T}(\hat{z}_T) \tag{5}$$

with the goal of making the autoencoder able to challenge the discriminator in the triplets mapping process. Equation (5) returns a high value when $\hat{\mathrm{D}}_\mathrm{T}$ recognizes mapped triplets as fake, and a low value when the encoder succeeds to deceive the discriminator.

At this point, we define our full objective for the training of the autoencoder model involving $\mathrm{M}$ and $\mathrm{M}^{-1}$ based on previous loss functions (1), (3), and (5) as:

$$\begin{aligned}\min_{\mathrm{M},\mathrm{M}^{-1}} J_{\mathrm{AE}}(\mathrm{M},\mathrm{M}^{-1}) = \mathbb{E}_{z_S\sim\mathcal{D}^S}\Big[&\lambda_R J_R\Big(z_S, \mathrm{M}^{-1}(\mathrm{M}(z_S))\Big)\\ &+ \lambda_T J_T\Big(\mathrm{M}(z_S); \hat{\mathrm{T}}_{\mathrm{T}}\Big) \quad (6)\\ &+ \lambda_D J_D\Big(\mathrm{M}(z_S); \hat{\mathrm{D}}_{\mathrm{T}}\Big)\Big]\end{aligned}$$

where each independent loss is weighted by a factor $\lambda_R$, $\lambda_T$ and $\lambda_D$ for balancing the effect of each loss function.

To enhance the discriminative power of $\hat{\mathrm{D}}_{\mathrm{T}}$, we opted to adopt the approach outlined in Algorithm 1 instead of directly optimizing Equation 6. By doing so, we enabled the simultaneous improvement of $\hat{\mathrm{D}}_{\mathrm{T}}$'s ability to distinguish between real and fake samples, while also minimizing the adversarial loss. The algorithm starts by taking the source dataset $\mathcal{D}^S$ and target dataset $\mathcal{D}^T$, as well as untrained models for $\hat{\mathrm{T}}_{\mathrm{T}}$, $\hat{\mathrm{D}}_{\mathrm{T}}$, $\mathrm{M}$ and $\mathrm{M}^{-1}$ as inputs. The first step employs the target dataset $\mathcal{D}^T$ to train $\hat{\mathrm{T}}_{\mathrm{T}}$ and $\hat{\mathrm{D}}_{\mathrm{T}}$ by solving optimization problems (2) and (4) respectively. This step provides a transition model to support the upcoming autoencoder training procedure and prepares the discriminator to recognize valid data from the original target dataset $\mathcal{D}^T$. Afterwards, the iterative training process starts by training the encoder and decoder models ($\mathrm{M}$ and $\mathrm{M}^{-1}$) solving optimization problem (6) and using pre-trained models $\hat{\mathrm{T}}_{\mathrm{T}}$ and $\hat{\mathrm{D}}_{\mathrm{T}}$. Encoder $\mathrm{M}$ is then used to transfer instances from the source dataset for the target task. The transferred dataset $\hat{\mathcal{D}}^T$ is then exploited to train the target discriminator model $\hat{\mathrm{D}}_{\mathrm{T}}$ along with the original target dataset $\mathcal{D}^T$. By iteratively upgrading the $\hat{\mathrm{D}}_{\mathrm{T}}$ ability to discern actual and generated target samples, we are also indirectly improving the encoder ability of mapping source samples into different, yet valid, regions of the target domain. The iterative training process continues until convergence, and the final result is the encoder and decoder components of the GAAN.

## IV. Experiments

In this section, we numerically evaluate our approach for learning the inter-task mapping and successively transforming triplets instances from a source to a target task. For all our validation experiments, we utilized the environments provided by the OpenAI Gym framework [2]. We chose as testing tasks three well-known benchmark environments in RL, namely the Mountain Car (MC) with continuous action, the Cart Pole (CP) and the Inverted Pendulum (IP), but our method may be extended to a diverse range of scenarios without any particular assumption.

### *A. Experimental Setup*

Starting from an available dataset of source triplets $\mathcal{D}^S$, we conducted two families of experiments. In the first one, described in Section IV-B, we evaluated the efficacy of the learned inter-task mapping by directly examining the influence of the transformed samples $\hat{\mathcal{D}}^T = \mathrm{M}(\mathcal{D}^S)$ on the prediction error of a transition model for the target environment. In the second one, described in Section IV-C, we further assess the performances of the mapping by testing the impact of the synthetically generated data $\hat{\mathcal{D}}^T$ in learning a target transition model employed as the environment simulator in the Deep Deterministic Policy Gradient (DDPG) [13] algorithm. This approach allows us to evaluate our inter-task mapping technique and its practical application in the context of Reinforcement Learning.

In the upcoming sections, we shall denote a transition model trained on $L$ triplets from the target dataset $\mathcal{D}^T$ and $S$ synthetic triplets from $\hat{\mathcal{D}}^T$ as $T_L^S$; in case $L$ or $S$ is equal to 0 the corresponding index is omitted. All the datasets contained unpaired triplets between the source and the target tasks, and they were gathered with a random exploration of their environments. To ensure a correct weighting of the independent components of each triplet in the datasets $\mathcal{D}^S$ and $\mathcal{D}^T$, we normalized the triplets before learning the mappings.

### *B. Model evaluation through prediction errors*

A core motivation behind the generation of synthetic triplets via the inter-task mapping is to feed such dataset $\hat{\mathcal{D}}^T$ into the learning process of a target transition model so as to improve its generalization capability.

**Algorithm 1** Instance Transferring through Generative Adversarial Autoencoding Network

```
Require:
 1: Source Dataset $\mathcal{D}^S$, Target Dataset $\mathcal{D}^T$, Target transition
    model $\hat{\mathrm{T}}_{\mathrm{T}}$, Target discriminator model $\hat{\mathrm{D}}_{\mathrm{T}}$, Autoencoder
    model AE
Ensure: Trained inter-task mapping $\mathrm{M}$ and $\mathrm{M}^{-1}$
 2: Train $\hat{\mathrm{T}}_{\mathrm{T}}$ solving optimization problem (2)
 3: Train $\hat{\mathrm{D}}_{\mathrm{T}}$ solving optimization problem (4) using only
    $\mathcal{D}^T$
 4: while not converged do
 5:     Train AE solving optimization problem (6)
 6:     $\mathrm{M} \leftarrow$ AE 's encoder
 7:     $\hat{\mathcal{D}}^T \leftarrow \mathrm{M}(\mathcal{D}^S)$
 8:     Train $\hat{\mathrm{D}}_{\mathrm{T}}$ solving problem (4) using $\mathcal{D}^T$ and $\hat{\mathcal{D}}^T$
 9: end while
10: return encoder $\mathrm{M}$
```

We started by training two transition models, namely $T_{200}$ and $T_{5000}$, on two MC target datasets $\mathcal{D}^T$ of 200 and 5000 triplets. The first model is used within Algorithm 1 to support the learning of the inter-task mapping, while the second one is conceived to act as a ground truth for our analysis. Next, starting from two datasets $\mathcal{D}^S$ of 5000 triplets coming from the source tasks IP and CP, we learnt with Algorithm 1 two inter-task mappings M, one from IP to MC, and the other one from CP to MC, and then we applied them to transform the source datasets into synthetic ones $\hat{\mathcal{D}}^T$ for the MC task. These transformed triplets were used, for each source task, to train two types of target transition models: one exclusively with the transformed data (respectively, $T\_pend^{5000}$ for the mapping IP to MC, and $T\_cp^{5000}$ for the mapping CP to MC), and the other one joining target and transformed triplets together (respectively, $T\_pend_{200}^{5000}$ and $T\_cp_{200}^{5000}$). Eventually, we trained an additional model $T\_multi_{200}^{10000}$, combining synthetic samples transformed from the two source environments IP and CP (5000 triplets each) and from the target MC (200 triplets).

To evaluate the models' prediction error, a test dataset $\mathcal{D}_{\text{test}}^T$ from the MC environment composed of 50000 triplets never seen during any phase of the algorithm or the learning of the target transition models was collected. For a model $T$, the prediction error was computed as $\|T(s_T, a_T) - s'_T\|_2^2$, with normalized triplets $(s_T, a_T, s'_T) \in \mathcal{D}_{\text{test}}^T$. Fig. 1a shows the error distribution for each trained model.

The results show that the errors produced by models trained only with synthetic data, $T\_pend^{5000}$ and $T\_cp^{5000}$ have a distribution similar to $T_{200}$, the transition model trained using only available target data. More importantly, the prediction error distribution of models employing both source and synthetic datasets, namely $T\_pend_{200}^{5000}$ and $T\_cp_{200}^{5000}$, is remarkably lower than the one of $T_{200}$, which is the model we would have had without the transfer technique. This result is even more evident for the model combining both source datasets transformed for the target task, that is $T\_multi_{200}^{10000}$.

We conducted further analysis to gauge the efficacy of the proposed inter-task mapping procedure and varied the sizes of the synthetic datasets $\hat{\mathcal{D}}^T$, ranging from 0 (i.e. no transfer at all) to 5000, repeating the process for three target datasets $\mathcal{D}^T$ of size: 100, 200 and 300 respectively. In all three cases, we observed a similar effect of the transfer as a function of the synthetic samples added during the training of the models. Fig. 1b and 1c show the prediction errors for the transformation of the synthetic datasets from the IP and CP source environments, respectively, over the test dataset $\mathcal{D}_{\text{test}}^T$, together with its 95% confidence interval over 20 repetitions of the experiment. In both scenarios, we can corroborate that the benefit provided by the transformed triplets is mostly observed for the very first synthetic samples and it remains nearly constant for larger amounts of additional samples. Finally, we fixed the amount of synthetic triplets and modified the number of actual target triplets available. This way, we could test the efficacy of our approach in contexts where the total amount of available data was different. As we expected, the average prediction error decreases with the amount of target triplets exploited. This result can be seen in Fig. 1d.

We have consistently observed that a target transition model trained with a sufficiently large collection of target triplets achieves the lowest prediction error, see $T_{5000}$ in Fig. 1. However, for tasks with low data availability, our method has proven to be an effective alternative, as shown in Fig. 1 for all the models trained with both synthetic and actual target data. It allows the model trained with both synthetic and target data to decrease its prediction error significantly when compared to models that use only the same amount of target triplets.

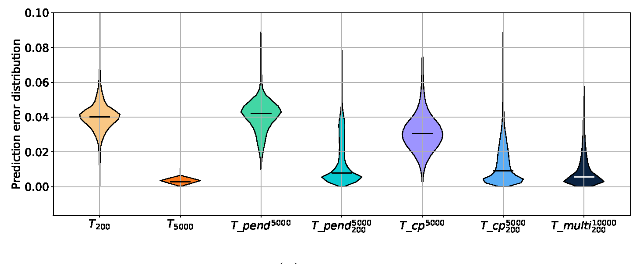
### *C. Model evaluation through Reinforcement Learning*

Transfer Learning aims to enhance the performance of RL control, minimizing the need of gathering additional information from the target task. In this study, we leveraged some of the models acquired in Section IV-B as simulators of the target environment. These simulators generate data that is utilized by an RL algorithm to learn a control policy. The performance of the learned policies is then evaluated in the actual target environment by quantifying the time to goal (i.e., the number of steps) for the MC problem. As a ground truth to compare our results, a control policy is also learned by the same RL algorithm by using the actual MC environment.

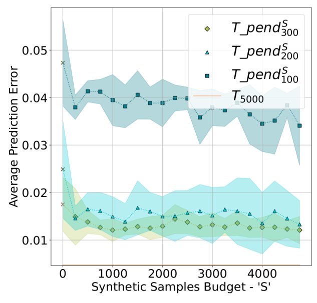
For our RL experiments, we adopted the DDPG algorithm [13] to learn the control policy , conducting 20 training iterations of 600 transitions each.

Fig. 2 reports the learning process of DDPG and the episode mean reward for each learning iteration. Since the accuracy of transition models affects the agent's reward during the learning, there are not obvious conclusions that can be derived by comparing each different curve in Fig. 2. However, we can observe a consistent increase in learning mean reward across most of the policies exploiting the simulators. Nevertheless, it appears that the policy trained with the model $T_{300}$, which was trained solely with a limited number of target triplets and without utilizing transfer learning, is the only one that fails to demonstrate effective learning.

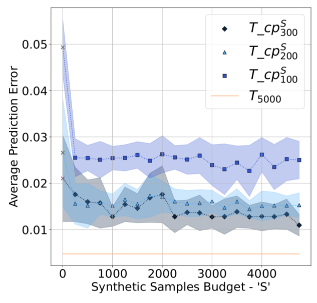
Once we've learned a control policy from each of the simulators, we test them in the actual MC environment. We reported in Fig. 3 an example of those policies for each simulator. As a performance index, here we use the number of steps the agent needs to reach the goal for each of 500 episodes with a maximum length 450. We can observe from both Fig. 2 and Fig. 3 that the model $T_{300}$ trained



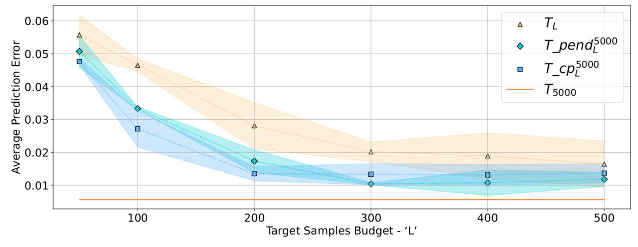
(a)



(b)



(c)



(d)

Fig. 1: Prediction error analysis for transition models trained for MC with different datasets. (a) Prediction Error distribution for relevant data configurations. (b) Effect on the prediction error of using a varying budget of synthetic samples mapped from IP, fixing the amount of target MC samples. (c) Effect on the prediction error of using a varying budget of synthetic samples mapped from CP, fixing the amount of target MC samples. (d) Effect on the prediction error of using different amount of target MC samples, fixing the amount of synthetic data.

using only 300 target triplets is not sufficiently accurate to learn an effective control policy. Nevertheless, our approach manages to exploit this small amount of target data, and learns a mapping that is able to transform additional samples from the source task into the target one, thus making the learning possible. For this kind of evaluation, $T\_pend^{5000}_{300}$ and $T\_cp^{5000}_{300}$ perform similarly to $T_{5000}$. Indeed the policies learned with $T\_pend^{5000}_{300}$ and $T\_cp^{5000}_{300}$ always reach the goal and the time to goal has a median similar to the policy learned

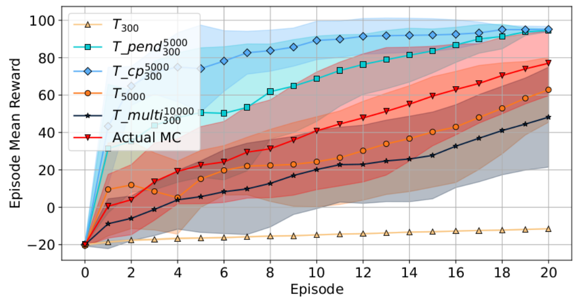


Fig. 2: Episode Mean Reward of DDPG Training for each trained transition model and for the actual Mountain Car. (5 runs, 95% c.i.)

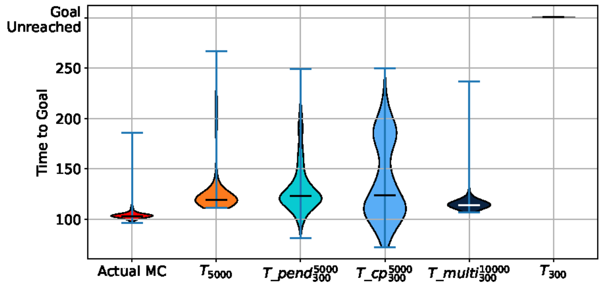


Fig. 3: Comparison between learned control policies over 500 episodes with maximum length of 450 steps.

through $T_{5000}$, even if they present an higher variance. By combining transformed data from multiple sources with the target ones, instead, performance are improved; policy learnt on $T\_multi^{10000}_{300}$ presents a decreased median and variance for the time to goal w.r.t. the policy learnt on any other model. One last observation can be made about the actual MC. Even though the actual MC is trained in the same test environment, it may occur that some of the policies learned through others simulators manage to reach the goal in a smaller number of steps. The rationale behind this behaviour can be traced back to the number of training iterations that we had to fix to have a fair comparison. But what is important to evaluate here, is the average time to goal, and the actual MC has the lowest median w.r.t. every simulator.

## V. Conclusions

This work proposes an adversarial approach to transfer knowledge from different domains through the generation of synthetic samples (instance transferring). The approach learns an inter-task mapping from a small amount of data of the target environment to transform source triplets into target ones, aiming to accurately replicate the target system dynamics and estimate an accurate transition model. In addition, our methodology facilitates the combination of knowledge coming from different source environments for a seamless extension to multi-source transfers. By exploiting the proposed method for inter-task mapping, we showed improvements in the generalization performance of RL algorithms, particularly when the original target data is insufficient to learn an accurate enough target transition model. The experimental results demonstrate how, through the transformed samples, it is possible to achieve higher prediction accuracy in the transition model of the target environment.

A limitation of the approach is its reliance on the transition model's quality utilized in Algorithm 1, which depends on the approximation architecture and the fitting algorithm employed. However, this particular aspect falls beyond the scope of our objectives in this study and should be investigated in future works. Moreover, in this work, we did not consider the potential applicability of the inverse inter-task mapping $\mathrm{M}^{-1}$, a side product of our methodology that can be further exploited to improve the quality of the instance transferring.

## Acknowledgements

This paper is supported by PNRR-PE-AI FAIR project funded by the NextGeneration EU program.

## References

[1] Ammar, H.B., Tuyls, K., Taylor, M.E., Driessens, K., Weiss, G.: Reinforcement learning transfer via sparse coding. In: Proceedings of the 11th international conference on autonomous agents and multiagent systems. vol. 1, pp. 383–390. International Foundation for Autonomous Agents and Multiagent Systems ... (2012)

[2] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv:1606.01540 (2016)

[3] Fernández, F., García, J., Veloso, M.: Probabilistic policy reuse for inter-task transfer learning. Robotics and Autonomous Systems **58**(7), 866–871 (2010)

[4] Garcke, J., Vanck, T.: Importance weighted inductive transfer learning for regression. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014. Proceedings, Part I 14. pp. 466–481. Springer (2014)

[5] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Communications of the ACM **63**(11), 139–144 (2020)

[6] Gupta, A., Devin, C., Liu, Y., Abbeel, P., Levine, S.: Learning invariant feature spaces to transfer skills with reinforcement learning. arXiv preprint arXiv:1703.02949 (2017)

[7] Kora, P., Ooi, C.P., Faust, O., Raghavendra, U., Gudigar, A., Chan, W.Y., Meenakshi, K., Swaraja, K., Plawiak, P., Acharya, U.R.: Transfer learning techniques for medical image analysis: A review. Biocybernetics and Biomedical Engineering **42**(1), 79–107 (2022)

[8] Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. AIChE journal **37**(2), 233–243 (1991)

[9] Lazaric, A.: Transfer in reinforcement learning: a framework and a survey. Reinforcement Learning: State-of-the-Art pp. 143–173 (2012)

[10] Lazaric, A., Restelli, M.: Transfer from multiple mdps. Advances in neural information processing systems **24** (2011)

[11] Lazaric, A., Restelli, M., Bonarini, A.: Transfer of samples in batch reinforcement learning. In: Proceedings of the 25th international conference on Machine learning. pp. 544–551 (2008)

[12] Lee, H., Battle, A., Raina, R., Ng, A.: Efficient sparse coding algorithms. Advances in neural information processing systems **19** (2006)

[13] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)

[14] Lu, Z., Zhu, Y., Pan, S., Xiang, E., Wang, Y., Yang, Q.: Source free transfer learning for text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 28 (2014)

[15] Mai, T., Yao, H., Zhang, N., He, W., Guo, D., Guizani, M.: Transfer reinforcement learning aided distributed network slicing optimization in industrial iot. IEEE Transactions on Industrial Informatics **18**(6), 4308–4316 (2022). https://doi.org/10.1109/TII.2021.3132136

[16] Niu, S., Liu, Y., Wang, J., Song, H.: A decade survey of transfer learning (2010–2020). IEEE Transactions on Artificial Intelligence **1**(2), 151–166 (2020)

[17] Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons (2014)

[18] Qureshi, A.S., Khan, A., Zameer, A., Usman, A.: Wind power prediction using deep neural network based meta regression and transfer learning. Applied Soft Computing **58**, 742–755 (2017). https://doi.org/https://doi.org/10.1016/j.asoc.2017.05.031

[19] Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)

[20] Taylor, M.E., Jong, N.K., Stone, P.: Transferring instances for model-based reinforcement learning. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II 19. pp. 488–505. Springer (2008)

[21] Taylor, M.E., Kuhlmann, G., Stone, P.: Autonomous transfer for reinforcement learning. In: AAMAS (1). pp. 283–290 (2008)

[22] Taylor, M.E., Stone, P.: Behavior transfer for value-function-based reinforcement learning. In: AAMAS. pp. 53–59 (2005)

[23] Taylor, M.E., Stone, P.: Cross-domain transfer for reinforcement learning. In: Proceedings of the 24th international conference on Machine learning. pp. 879–886 (2007)

[24] Taylor, M.E., Stone, P., Liu, Y.: Transfer learning via inter-task mappings for temporal difference learning. Journal of Machine Learning Research **8**(9) (2007)

[25] Taylor, M.E., Whiteson, S., Stone, P.: Transfer via inter-task mappings in policy search reinforcement learning. In: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems. pp. 1–8 (2007)

[26] Tirinzoni, A., Salvini, M., Restelli, M.: Transfer of samples in policy search via multiple importance sampling. In: International Conference on Machine Learning. pp. 6264–6274. PMLR (2019)

[27] Tirinzoni, A., Sessa, A., Pirotta, M., Restelli, M.: Importance weighted transfer of samples in reinforcement learning. In: International Conference on Machine Learning. pp. 4936–4945. PMLR (2018)

[28] Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. Journal of Big data **3**(1), 1–40 (2016)

[29] Yang, Y., Guo, J., Ye, Q., Xia, Y., Yang, P., Ullah, A., Muhammad, K.: A weighted multi-feature transfer learning framework for intelligent medical decision making. Applied Soft Computing **105**, 107242 (2021)

[30] You, H., Yang, T., Zheng, Y., Hao, J., E Taylor, M.: Cross-domain adaptive transfer reinforcement learning based on state-action correspondence. In: Uncertainty in Artificial Intelligence. pp. 2299–2309. PMLR (2022)

[31] Zhang, Q., Xiao, T., Efros, A.A., Pinto, L., Wang, X.: Learning cross-domain correspondence for control with dynamics cycle-consistency. arXiv preprint arXiv:2012.09811 (2020)

[32] Zhao, L., Pan, S., Xiang, E., Zhong, E., Lu, Z., Yang, Q.: Active transfer learning for cross-system recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 27, pp. 1205–1211 (2013)

[33] Zhao, W., Queralta, J.P., Westerlund, T.: Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 737–744 (2020). https://doi.org/10.1109/SSCI47803.2020.9308468

[34] Zhu, Z., Lin, K., Zhou, J.: Transfer learning in deep reinforcement learning: A survey. arXiv preprint arXiv:2009.07888 (2020)