Cross-modal Domain Adaptation for Cost-Efficient Visual Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

In visual-input sim-to-real scenarios, to overcome the reality gap between images rendered in simulators and those from the real world, domain adaptation, i.e., learning an aligned representation space between simulators and the real world, then training and deploying policies in the aligned representation, is a promising direction. Previous methods focus on same-modal domain adaptation. However, those methods require building and running simulators that render high-quality images, which can be difficult and costly. In this paper, we consider a more cost-efficient setting of visual-input sim-to-real where only low-dimensional states are simulated. We first point out that the objective of mapping functions in previous methods that align the representation spaces is ill-posed, prone to yield an incorrect mapping. When the mapping crosses modalities, previous methods are easier to fail. Our algorithm, **C**ross-m**O**dal **D**omain **A**daptation with **S**equential structure (CODAS), mitigates the mismatching problem by utilizing the sequential nature of the data sampling process in RL tasks. Experiments on MuJoCo and Hand Manipulation Suite tasks show that the agents deployed with our method achieve similar performance as it has in the source domain, while those deployed with previous methods designed for same-modal domain adaptation suffer a larger performance gap.

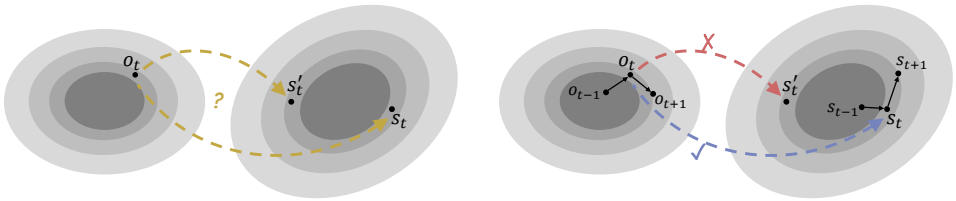
1 Introduction

Reinforcement learning (RL) for vision-based robotic control tasks has achieved remarkable success in recent years [1, 2]. However, current RL algorithms necessitate a substantial number of interactions with the environment, which are costly both in time and money on real robots. An appealing alternative is to train policies in simulators, then transfer these policies to real-world systems [3]. Due to inevitable differences of representation between simulators and the real world, which is also known as the "reality gap" [4], applying policies trained in one domain directly to another almost surely fail, especially in visual-input tasks, which is due to the poor generalization of RL policies [5]. Domain adaptation is a promising direction to handle the gap by mapping representation from two domains to an aligned representation and then training and deploying policies in the aligned representation.

Many recent works have adopted unsupervised visual domain adaptation [6, 7, 8], which learn a mapping function to align the the data distributions of the two domains. We point out that, as illustrated in Fig. 1(a), the objective is ill-posed for learning a correct mapping function. These adaptation methods exploit structural constraints [9] in two domains of the same modality (e.g., learned on simulated **images** and deployed on real **images**). These methods implicitly alleviate the intrinsic ill-posedness of distribution matching.

However, such kind of same-modal domain adaptation requires the simulator to render images when training the model, which introduces unwanted costs and difficulties that are ignored in previous

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) Mapping only considering distribution matching (b) Mapping with sequential structure

Figure 1: Illustration of the training objectives for mapping function in unsupervised domain adaptation. Shaded regions denote data distributions, where the darker the color, the higher the probability. For each figure, the left region is the target domain and the right is the source domain. In Fig. 1(a), both s_t and s'_t are "realisic" instances, but only s_t is correct. Since they are of similar probabilities, mapping an instance o_t in the target domain to somewhere of a similar probability in the source domain is "reasonable" if we only consider distribution matching. In RL, the policy may output unreliable actions when taking these incorrectly mapped states as inputs. In Fig. 1(b), a sequential structure can help rule out the wrong mapping via trajectory contexts.

works. First, a rendering engine needs costly human engineering. Second, using RL methods to train a policy with an image-based simulator is usually harder [10] and slower (can be up to 20× slower [11]) than with a state-based simulator. An ideal solution to these problems is to train policies with states in the simulator and adapt the learned policies to real-world images. Current domain adaptation methods generally fail on this setting since the structural constraints based on the modality consistency are no longer available, which makes the representation alignment task harder.

To learn such a cross modal mapping, we propose Cross-mOdal Domain Adaptation with Sequential structure (CODAS) that learns a mapping function from images in the target domain to states in the source domain, as illustrated in Fig. 1(b). With the help of the learned mapping function, policies trained on states in the source domain can be deployed in the target domain of images directly. Specifically, based on the sequential nature of RL problems, we formulate the cross-domain adaptation problem as a variational inference problem and reformulate them into a series of solvable optimization objectives. We also design a special residual model structure in the recurrent neural network (RNN) to enforce additional inductive bias and stabilize the training process.

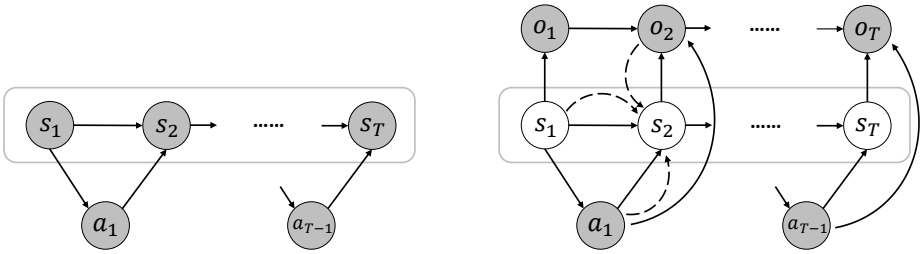
We evaluate our method on six MuJoCo [12] tasks and four Robot Hand Manipulation tasks [13], where we treat states as the source domain, and rendered images as the target domain. Results show that the learned mapping function can help transfer the policy to the target domain with only a small performance degradation in most of the tasks, while existing methods [14, 15] suffer a larger performance gap.

2 Related Work

Unsupervised visual domain adaptation (UDA) aims to map the source domain and the target domain to an aligned distribution without pairing the data. UDA is originally designed for image translation in computer vision [14, 6, 16, 17, 18]. In RL, UDA is used to transfer policies from simulators to the real world, hoping to overcome the reality gap between rendered images and real world images. Previous methods fall into two major categories: feature-level adaptation, where domain-invariant representations are learnt [7], and image-to-image adaptation, where pixels from source images are used to generate images that look like those from the target domain [19].

Feature-level adaptation usually adopts domain randomization techniques for domain-invariant representation learning [20, 21, 22, 23, 24]. In domain randomization, a meta-simulator is required to generate a mass of variants on rendered images. The learner aims to extract an invariant representation from the variants of representations. Therefore, the training process is generally costly. Besides, these methods implicitly assume that the variants can cover the target domain, so that the target domain is only an instance of the variants.

Image-to-image adaptation is challenging when data from two domains are unpaired. Most previous works attempt to solve this problem by using generative adversarial networks (GANs) [15, 25, 3, 26, 27, 28]. [25] transfers policies from Atari games to their modified variants by training a GAN to map



(a) Generation process in the source domain

(b) Generation and inference process in the target domain

Figure 2: Illustration of the generation and inference processes based on the decision processes of RL tasks. All nodes are random variables. Shaded nodes are observable variables. Solid lines denote the generation process and dashed lines denote the inference process. There is an correspondence between two state trajectories surrounded which is denoted by the rounded rectangles. Note that we include policy π in both generation processes, which correspond to the edge from state s_t to action a_t.

images from the target domain to the source domain. RL-CycleGAN [3] unifies the learning of a CycleGAN [14] and an RL policy, claiming better performance by learning features that are most crucial to the Q-function in RL.

Despite the success of the image-to-image domain adaptation methods in previous settings, we point out that these methods can somewhat bypass the ill-posedness for distribution matching, which is illustrated in the example in Fig. 1. Since images generally differ only locally in color, textile, and lighting, but resemble globally between two domains, the effect of ill-posedness can be implicitly handled by leveraging the consistency of modality [26, 27, 28]. However, since images and states differ essentially, we can no longer utilize the consistency of modality. Thus, as shown in our experiments, the effect of ill-posedness is revealed in RL tasks. Some works impose extra structural constraints [9, 29], but fail in image-to-state domain adaptation either. In our work, we force the mapped states to follow transition consistency by considering the inner relationship between the sampled data from the decision-making process in RL tasks.

3 Cross-modal Domain Adaptation with Sequential Structure

Without loss of generality, the visual domain adaptation problem in RL can be formulated as learning a mapping function to align the representation of the source and target domains, then training and deploying policies in the aligned representation. To learn the mapping function and policies, a dataset is pre-collected by some policies (e.g., human-expert policies or random policies) in the target domain and an agent can only interact with the environment in the source domain. Since our major concern is the reality gap between the representation spaces, in this formulation, we follow the setting in previous methods which omits the gap between the dynamics models in the two domains.

In our setting of cross-modal domain adaptation, the source domain is low-dimensional state domain and the target domain is high-dimensional image domain. In this work, we regard the state space as the aligned representation space. We first pre-train a policy π in the source domain. By learning a mapping q_ϕ, parameterized by ϕ, from image space O to state space S, the agent can be deployed with a new policy $\tilde{\pi}(o) = (\pi \circ q_\phi)(o)$, where $\circ$ denotes function composition, i.e., image observations are mapped to states, then the pre-trained policy directly acts on the states.

3.1 Domain Adaptation in RL as Variational Inference

In this section, we first define our objective of domain adaptation in RL based on the framework of variational inference. Then we give a comparison between our objective and the objectives which are modeled in previous methods.

We first define the generation and inference processes in the source and target domains in the decision process of RL. The illustration is given in Fig. 2. In the source domain, an initial state follows the distribution $p(s_1)$. A transition function $s_t \sim p(s \mid s_{t-1}, a_{t-1})$ gives the current state s_t from the

previous state s_{t-1} and the previous action a_{t-1}. In the target domain, the generation process also follows the transition of states, but the states are not available to the agent and the agent only observes the images corresponding to the states. An image observation o_t is generated by the corresponding state s_t. In other words, there exists a decoder $p(o_t \mid s_t)$ that can construct observation from state. In practice, our decoder is defined as $p(o_t \mid s_t, o_{t-1})$. We allow $p_\theta(o_t \mid \cdot)$ to be dependent on o_{t-1}, so that some irrelevant patterns in images can be decoded easily in an auto-regressive manner. The inference process can be regarded the mapping process in domain adaptation in RL. The inference function is defined as $q_\phi(s_t \mid s_{t-1}, a_{t-1}, o_t)$, which is similar to the transition function but with additional o_t as an input. A simple insight is that the inputs s_{t-1} and a_{t-1} give information to predict the distribution of s_t, as the transition function $p(s \mid s_{t-1}, a_{t-1})$ does, and o_t is used to determine the state sample s_t from the distribution.

The training process aims at learning an inference function (i.e., the mapping function) that approximates the ground-truth distribution with a posterior distribution. The raw optimization objective of variational inference in this setting can be formulated as:

$$\min_{\phi} \mathbb{E}_{\tau^o} \left[D_{\mathrm{KL}} \left[q_\phi(\tau^s \mid \tau^o) \mid\mid p(\tau^s \mid \tau^o) \right] \right], \tag{1}$$

where τ denotes a trajectory, the superscripts s and o of τ indicate the trajectories from the source and target domains respectively, $p(\cdot)$ is the ground-truth distribution, $q_\phi(\cdot)$ is the mapping function that we want to learn, and D_{KL} computes the Kullback-Leibler divergence. τ^o contains a trajectory of observation-action pairs $\{(o_1, a_1), (o_2, a_2), ..., (o_T, a_T)\}$, and τ^s contains a trajectory of state-action pairs $\{(s_1, a_1), (s_2, a_2), ..., (s_T, a_T)\}$. Here we assume the trajectories of source and target domains are collected by the same policy π, which is a mild assumption and has been implicitly or explicitly introduced in previous works [25]. Since the policy is fixed, for the simplification of notations, we omit the dependence of π and just use q_ϕ and p to indicate the distribution. The Evidence Lower BOund (ELBO) of this variational problem can be formulated as:

$$\max_{\phi,\theta} \mathbb{E}_{\tau^o} \Big[\mathbb{E}_{\hat{\tau}^s \sim q_\phi(\tau^s \mid \tau^o)} \left[\log p_\theta \left(\tau^o \mid \hat{\tau}^s \right) \right] - D_{\mathrm{KL}} \left[q_\phi \left(\tau^s \mid \tau^o \right) \mid\mid p \left(\tau^s \right) \right] \Big], \tag{2}$$

where $\hat{\tau}^s$ denotes the inference trajectories of q_ϕ, and $p_\theta(\tau^o \mid \hat{\tau}^s)$ is an approximation of the generation process $p(\tau^o \mid \tau^s)$, with parameter θ. The derivation of ELBO can be found in Appendix A. The first term maximizes the reconstruction probability, in order to enforce that the mapped state $\hat{s}$ can recover observation o. The second term enforces the alignment of the distributions of the mapped trajectories $q_\phi \left(\tau^s \mid \tau^o \right)$ and the ground-truth trajectories distribution $p \left(\tau^s \right)$ in the source domain.

Compared with previous works which learn the mapping function by minimizing the divergence between the distributions of $q_\phi(s \mid o)$ and $p(s)$, we model the optimization objective as a matching of trajectory distributions via the Bayesian graphical model. As illustrated in Fig. 1, the trajectory-level modeling is important since the data points are not i.i.d., since they are collected in the RL scenario. Without considering the inner structure of the dataset, mismatching of mapping might occur easily.

3.2 Differentiable Optimization Objectives

The ELBO defined in Eq. (2) contains terms involving distributions over the entire trajectory, which is impractical to solve. Given the previously defined generation process, we can decompose the joint probability into the multiplication of single-step probabilities. The result of the decomposition is

$$\max_{\phi,\theta} \mathbb{E}_{\tau^o \sim \mathcal{D}^o} \Big[\sum_{t=1}^{T} \mathbb{E}_{\hat{s}_t \sim q_\phi(s_t \mid s_{t-1}, a_{t-1}, o_t)} \big[\log p_\theta(o_t \mid \hat{s}_t, o_{t-1}, a_{t-1}) \big] - D_{\mathrm{KL}} \left[q_\phi(\tau^s \mid \tau^o) \mid\mid p(\tau^s) \right] \Big], \tag{3}$$

where $\mathcal{D}^o$ denotes the pre-collected dataset in the target domain, t is the timestep and T denotes the horizon. s_0, a_0 and o_0 are initialized with $\mathbf{0}$. $q_\phi(s_t \mid s_{t-1}, a_{t-1}, o_t)$ is an RNN which outputs the mean and standard deviation of a Gaussian distribution. A direct computation of the second D_{KL} term in Eq. (3) is intractable. Following the idea that the optimization process of GAN is equivalent to minimizing a certain distance measurement between two distributions [30], we use the optimization objective of GAN as the alternative of minimizing $D_{\mathrm{KL}} \left[q_\phi \left(\tau^s \mid \tau^o \right) \mid\mid p \left(\tau^s \right) \right]$. For the simplicity of implementation and stability of training, we use the original optimization objective of GAN, which is equivalent to minimizing the Jensen-Shannon divergence. The optimization objective is

$$\max_{\omega} \min_{\phi} \mathbb{E}_{\tau^s \sim \mathcal{D}^s} \big[\log \left(D_\omega(\tau^s) \right) \big] + \mathbb{E}_{\tau^o \sim \mathcal{D}^o, \hat{\tau}^s \sim q_\phi(\tau^s \mid \tau^o)} \big[\log \left(1 - D_\omega(\hat{\tau}^s) \right) \big]. \tag{4}$$

Similarly, we decompose Eq. (4) into single-step formulation:

$$\max_{\omega}\min_{\phi} L_D(\omega,\phi) = \max_{\omega}\min_{\phi} \mathbb{E}_{\tau^s\sim\mathcal{D}^s}\big[\sum_{t=1}^{T}\log D_\omega(s_t,a_t,h_{t-1})\big] \\ +\mathbb{E}_{\tau^o\sim\mathcal{D}^o}\big[\sum_{t=1}^{T}\mathbb{E}_{\hat{s}_t\sim q_\phi(s_t|s_{t-1},a_{t-1},o_t)}\log\big(1-D_\omega(\hat{s}_t,a_t,h_{t-1})\big)\big], \tag{5}$$

where $\mathcal{D}^s$ denotes the dataset collected by the pre-trained policy in the source domain. Here D_ω is also implemented as an RNN, in which h denotes the hidden state in the RNN rollout. In particular, the discriminator outputs a tuple $(y_t, h_t) = D_\omega(s_t, a_t, h_{t-1})$, where y_t is the probability of prediction and h_t is the next hidden state. We omit the output of h for brevity. The latter term is the practical objective of the KL-divergence in Eq. (3) of the mapping function q_ϕ. This part is different from VAE-GAN [31] which discriminates against the reconstructed output. Instead, the discriminator discriminates against the states outputted by the variational encoder.

Combining both the aforementioned reconstruction loss and generation loss together, the optimization objective of the mapping function is

$$\max_{\phi,\theta}\mathbb{E}_{\tau^o\sim\mathcal{D}^o}\Big[\sum_{t=1}^{T}\mathbb{E}_{\hat{s}_t\sim q_\phi(s_t|s_{t-1},a_{t-1},o_t)}\big[\log p_\theta(o_t\mid\hat{s}_t,o_{t-1},a_{t-1}) \\ -\lambda_D\log\big(1-D_{\omega^*}(\hat{s}_t,a_t,h_{t-1})\big)\big]\Big], \\ \text{s.t. } \omega^* = \arg\max_{\omega} L_D(\omega,\phi), \tag{6}$$

where λ_D is the hyper-parameter for the weight controlling. The decoder p_θ and the discriminator D_ω are fixed during the training process of the mapping function q_ϕ. The loss function of the decoder p_θ is the first term in Eq. (6). Similarly, the mapping function q_ϕ is fixed during the training process of the decoder p_θ and discriminator D_ω. The detailed derivation can be found in Appendix A.

3.3 Stablizing the Inference Function

Trajectories of practical RL problems often last hundreds of time steps. Training RNN on such long-horizon trajectories can be difficult. The previous success of ResNet [32] shows that a residual structure can simplify the learning target to predicting small residuals, resulting in a remarkable performance increase in learning ultra-deep neural networks.

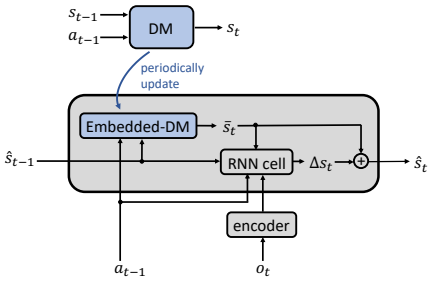


Figure 3: Model structure with embedded-DM.

Adopting a similar idea, we incorporate a residual structure in the RNN to help stabilize the training process. A deterministic Dynamics Model (DM) $p_\varphi(s,a)$ with parameter φ is trained independently using transition tuples collected in the simulator. Its parameters are periodically updated to embedded-DM. DM minimizes the mean-square error to the transition tuples in the source domain. We regard the embedded-DM as a part to provide an "average" estimation of the next states $\bar{s}_t$ (see Fig. 3). The job of the rest of the part, instead, is simplified to just output a correction. With this specially designed model structure, the mapped states follow the transition dynamics in the source domain better. In particular, we model the mapping function as:

$$\hat{s}_t = p_\varphi(s_{t-1},a_{t-1}) + \alpha\Delta s_t, \Delta s_t \sim q_\phi(\Delta s \mid s_{t-1},a_{t-1},o_t), \tag{7}$$

where α is a hyper-parameter to control the correction range of Δs. Δs takes tanh as the activation function. Thus, the range of $\alpha\Delta s$ is constrained to $[-\alpha,\alpha]$.

DM is first trained using batches of transition tuples collected in the simulator. However, these data are insufficient for DM training since the DM predictions in unseen state-action pairs are unreliable, which is called extrapolation error [33, 34]. Since the $\alpha\Delta s$ is constrained to $[-\alpha,\alpha]$, too larger extrapolation

errors of $p_\varphi(s_{t-1}, a_{t-1})$ might obstruct Δs_t to recover the correct $\hat{s}_t$. During the training process of the mapping function, the dynamics model is trained online using $\mathcal{D}^{\hat{s}} = \{(\hat{s}, \pi(a \mid \hat{s}), p(\hat{s}, \pi(a \mid \hat{s}))\}$. That is, we reset the simulator to the mapped states $\hat{s}_t$ and then rollout with a single-step oracle transition to get s_{t+1}. The optimization objective of the DM is to minimize the following mean-squared-error loss:

$$\min_{\varphi} \mathbb{E}_{(s,a,s') \sim D^s \cup D^{\hat{s}}} [(p_\varphi(s, a) - s')^2]. \tag{8}$$

When training the mapping function q_ϕ, the parameter φ in the embedded-DM is fixed and gradients are backpropagated from $\hat{s}$ in Eq. (7) to ϕ via the reparametrization trick [35]. The detailed training procedure of CODAS is shown in Algs. 1 and 2 in the Appendix.

4 Experiments

We evaluate our method in MuJoCo environments [12] from OpenAI Gym and Robot Hand Manipulation Tasks. We define the rendered images as the target domain, and the original states as the source domain. The pre-collected dataset in the target domain contains 600 trajectories and is collected by policies trained by PPO [36] and DAPG [37]. Please refer to Appendix D for the detailed experimental setting. The quantitative performance of the data-collecting policies are given in Appendix E.2.

We compare our method with several state-of-the-art methods in same-modal domain adaptation which are also compatible to cross-modal setting, including:

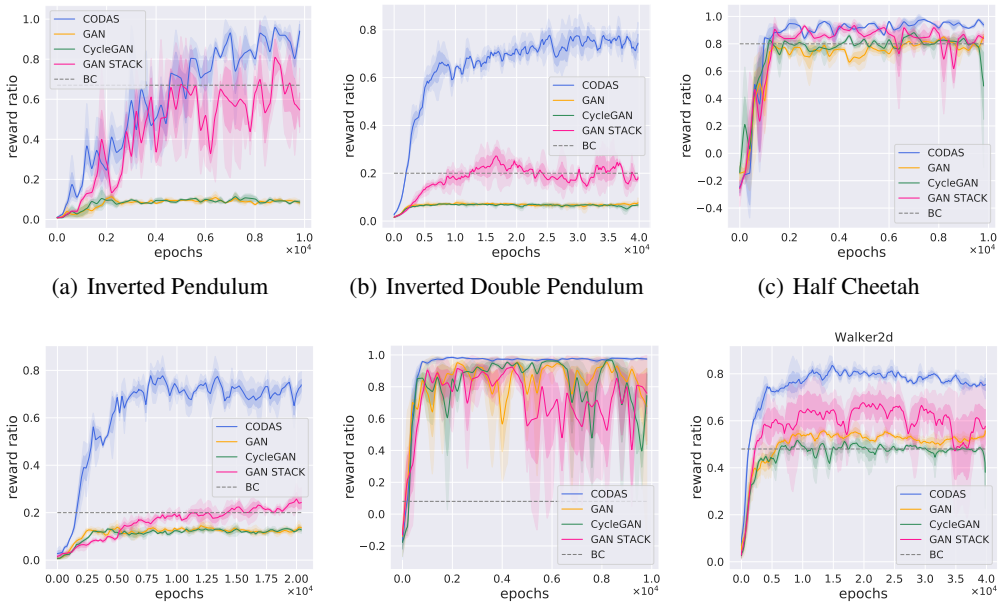
1. GAN: Conditional GAN trains a mapping function $p_\phi(s \mid o)$ to align the distributions of $\mathcal{D}^o$ and $\mathcal{D}^s$. A discriminator tries to discriminate the mapped states from those from D^s, and is used to guide the training of the mapping function.
2. CycleGAN: CycleGAN shares the same basic framework with GAN with an extra cycle-consistent loss to constrain the mapping. In other words, CycleGAN generates $\hat{o}$ from $\hat{o} \sim p_{\phi'}(o \mid p_\phi(s \mid o)))$ and trains the mapping function via making the $D(\hat{o})$ output the real labels with high probability.
3. GAN with Stacked Input: Inspired by a common trick that uses stacked images as input in visual RL tasks [38, 39], we modify GAN to take stacked images as input, and name it GAN STACK. The stacked images provide additional local temporal information.

We also compare CODAS with behavioral cloning (BC), which trains a policy in a supervised manner using $(o_t, a_t) \sim \mathcal{D}^o$. All domain-adaptation baselines are trained with the same neural network structure and tuned to converge (i.e., the output probabilities of discriminators are close to 0.5). The tasks are trained for 10,000 to 40,000 epochs based on their difficulty. For each epoch, every method is updated using a batch of 20 trajectories. Implementation details for all these methods can be found in Sec. C.

To focus on the performance of the adaptation process, we use reward ratio as the metric: $r_{\text{ratio}} = \frac{r}{r^*}$, where r and r^* are the cumulative rewards of the adapted policy deployed on images the optimal policy trained on states respectively. Each tasks are trained with three seeds. In the following plotted curves, the solid lines indicate the mean value and the shaded regions indicate the range of ± 1 standard deviation.

4.1 Performance on MuJoCo Tasks

The training curves of all methods in MuJoCo tasks are given in Fig. 4. Firstly, BC performs well in HalfCheetah, but poorly in the other environments. This indicates that the dataset is not large enough to learn a correct mapping from images to actions directly. In Swimmer and HalfCheetah, the deployed policies with mapping functions learned by GAN and CycleGAN achieve good performances. It is because of the stability of the environments. The trajectory will not pre-terminate no matter how badly the agent behaves. Therefore, the agent is allowed some incorrect actions, which come from mismatched states. While in the rest tasks, the agent might reach unsafe states after performing some undesired actions. Thus the agent is sensitive to the mismatching of mapping states. In these tasks, GAN and CycleGAN peform even worse than BC. Although CycleGAN has been tuned to converge (see Appendix), the performance of CycleGAN is similar to GAN in all of the tasks. We believe although the cycle-consistent loss guarantees the mapping function is a bijection mapping



(a) Inverted Pendulum (b) Inverted Double Pendulum (c) Half Cheetah

(d) Hopper (e) Swimmer (f) Walker2d

Figure 4: Training curves of different methods on MuJoCo.

(i.e., one-to-one mapping), the mismatching problem shown in Fig. 1(a) can not be handled, which is more critical in the RL setting. Therefore the performance improvement is not significant. Finally, we found that the GAN STACK algorithm reaches better performances than GAN in most of the tasks. The phenomenon indicates that sequential information is important to learn a correct mapping. However, without other constraints as CODAS has, the performance improvement is more unstable.

The performance of CODAS is consistently better than the baseline algorithms. Besides, the performance of deployed policies is around 85% of the original performance. A visual illustration of the mapped states' accuracy is shown in Fig. 5. Both reconstructed images and re-rendered images match the original ones well. Re-rendered images can even match the original ones well for the last falling frames which are sparse in the dataset.

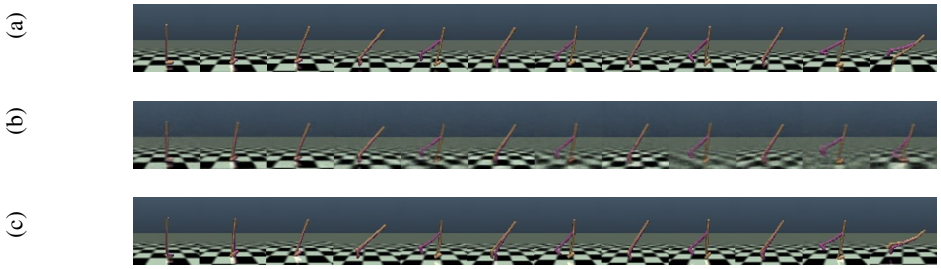
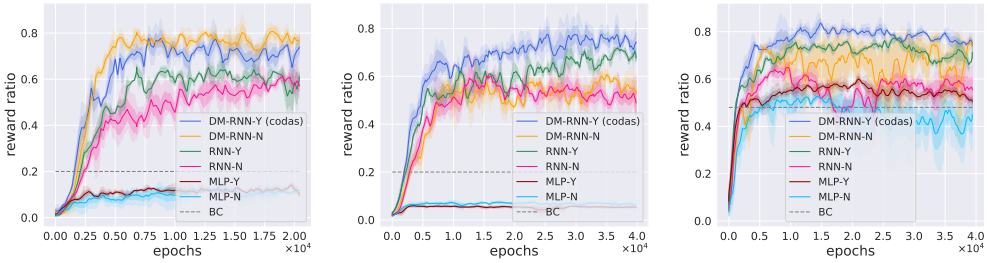


Figure 5: A visual illustration of (a) original images, (b) reconstructed images, and (c) re-rendered images of the mapped states.

4.2 Analysis of CODAS

By considering three mapping function modeling methods (i.e., Multilayer Perceptron (MLP), RNN or RNN with embedded-DM (DM-RNN)) and two discriminator modeling methods (i.e., taking trajectories as inputs or not), we construct six CODAS variants to do our Ablation studies. We named them with the format of "mapping method-Y/N". For example, `DM-RNN-Y` is the algorithm that the mapping function is modeled with RNN and embedded-DM, and the discriminator taking

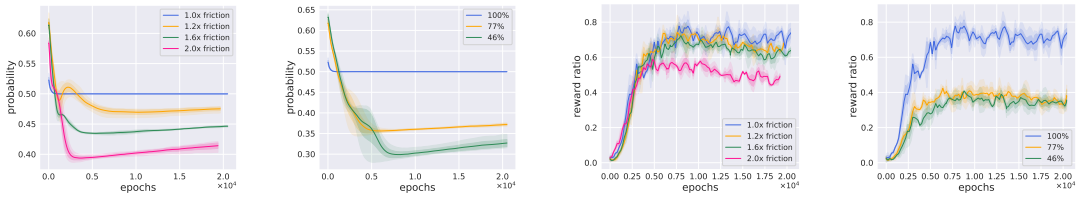
trajectories as inputs, which is the original CODAS. We conduct experiments in Hopper, Inverted Double Pendulum and Walker2d. We show the results in Fig. 6.



(a) Hopper (b) Inverted Double Pendulum (c) Walker2d

Figure 6: Ablation studies of CODAS.

The comparison of `RNN-*` methods and `MLP-*` methods in Fig. 6 shows that RNN structure is an important component for mapping function training in Hopper and Inverted Double Pendulum tasks. By taking sequential information into consideration, the actions taken in the target domain are more accurate because the policy acts on the mapped states. Based on the RNN mapping function, feeding the sequential information into the discriminator further improves the performance for InvertedDouble and Walker2d. These results verify our argument that the inner structure of the collected dataset is crucial for modeling the universal domain-adaptation objective in RL setting. On the other hand, in these tasks, embedded-DM helps improve the performance both from `RNN-N` to `DM-RNN-Y` and from `RNN-Y` to `DM-RNN-Y`. Both embedded-DM and the sequential-input discriminator improve the performance of CODAS from two aspects: embedded-DM standardize the inference process so that to simplify the learning complexity; the sequential-input discriminator strengthens the gradient signals back-propagated from the discriminator by taking more information into consideration to discriminate the two datasets.



(a) Discriminator prediction (dynamics mismatch) (b) Discriminator prediction (policy mismatch) (c) Performance (dynamics mismatch) (d) Performance (policy mismatch)

Figure 7: The training curves of CODAS under two types of mismatches. Y-axis in Fig. 7(a) and Fig. 7(b) is the mean of predicted probabilities of the discriminator given a set of trajectories with mapped states. The higher probability means the data are more like to be real. In the dynamics mismatch setting, we use "number-x" to denote the rescaling coefficient to the original friction coefficient. For example, "1.2x" denotes the friction coefficient in the source domain is 1.2 times to the target domain. In the policy mismatch setting, the percentage is to denote the performance ratio of the target domain policy compared with the policy in the source domain.

Current UDA methods omit the gap between the dynamics models and the data collected policies of the source and target domains [26, 27]. Although we follow these implicit consistency assumptions in CODAS, in some real-world applications, the reality-gap of dynamics models is non-negligible [20, 40, 41, 42], and the policies in two domains might not be the same. Unfortunately, whether considering the reality-gap of dynamics models or the mismatching in the policies, the primitive objective of CODAS and previous UDA methods are biased: Even with an oracle mapping function $p(s \mid o)$, the distribution of the dataset in source domain D^s and the mapped dataset $D^{\hat{s}} := \{\hat{s} \mid \hat{s} \sim p(s \mid o), o \in D^o\}$ are not aligned [43]. In this section, we analyze the effect of the violation on the implicit consistency assumptions in current UDA paradigm.

We first change the friction coefficient in the Hopper environment to analyze the effect of dynamics mismatches to CODAS. As shown in Fig. 7(a), with a larger friction difference, the asymptotic

probability tends to be smaller. The phenomenon indicates that the gap between dynamics models can lead to the gap between the state-action distributions, which can be distinguished by the discriminator. When the rescaling ratio of friction is larger than 2.0, the gap between state-distributions starts to affect the deployment performance of policy (Fig. 7(c)). As shown in Fig. 7(b), the mismatch of collected policies also induces the gap of the state-action distribution. The gap is larger compared with that of the dynamics mismatch. Such a larger gap do cause a more significant performance drop of deployed policy.

However, In the experiments with 120% and 160% friction, the mapping function proves to be more robust to the mismatch. We think that such tolerance comes from the variational inference modeling of CODAS, which introduces distribution modeling for each step of inference. The distribution modeling improves the robustness by considering the uncertainty of the prediction. The performance drop in both experiments suggest us pay more attention to the implicit assumptions that have not been fully discussed in current UDA paradigm.

4.3 Performance on Complex Robot Hand-Manipulation Environments

We further test CODAS in four complex hand-manipulation tasks: door, pen, hammer and relocate. A visualization of the hammer task is given in Fig. 8. In the hammer task, the robot arm should hold the hammer and hit the nail into the wood. We use the pre-trained polices provided by these tasks,[1] which are trained with DAPG. The performance of CODAS is given in Tab. 1. In three out of the four tasks, CODAS yields reasonable mapping functions for policy deployment. CODAS fails to output a correct mapping in the relocate task. We believe the failure comes from the goal-conditional nature of the task, in which the robot arm should grasp a ball and take it to a randomly initialized target location in every trajectory. The goal-conditional nature increases the complexity to learn a good policy and requires a larger dataset to learn a mapping function.

Figure 8: Visualization of the hammer task.

Table 1: The reward ratio in hand-manipulation tasks.

Tasks	hammer	pen	door	relocate
Reward Ratio	0.820	0.701	0.886	0.090

5 Discussion and Future Work

In this work, we investigate the cross-modal unsupervised domain adaptation problem, which we believe is more cost-efficient than image-to-image domain adaption. We first point out the intrinsic ill-posedness of distribution matching in the current UDA objective formulation. To handle the issue, we model our objective of domain adaptation based on the decision processes of RL tasks and the framework of variational inference, then derive a differentiable and stabilizing objective. Finally, we propose Cross-Modal Domain Adaptation with Sequential structure (CODAS). The results in MuJoCo and Robot Hand Manipulation Tasks show the ability of CODAS to find a correct mapping function without pre-labeling the data. In the ablation studies, we analyze the effectiveness of sequential mapping structure, trajectory-input discriminator, and embedded-DM in CODAS. We also point out the limitation of CODAS and the current UDA paradigm: the omitted implicit consistency assumptions on dynamics models and data-collected policies, which might not be satisfied in some applications, would make the primitive objective of current UDA methods are biased. The bias of the objective might hurt the deployed performance.

Exploiting more efficient structure constraints to the specific scenarios to stabilize mapping function training is effective to solve the UDA problem, as current methods do. But we expect to inspire the researchers that modeling the process of mapping and generation more elaborately is also a potential way to improve the performance on UDA. By incorporating the mismatching of dynamics models and data-collected policies into the CODAS framework, we can build a more robust UDA algorithm when the assumptions in this paper are not strictly satisfied. We leave this as our future work.

[1] https://github.com/aravindr93/hand_dapg

References

[1] Anthony G. Francis, Aleksandra Faust, Hao-Tien Lewis Chiang, Jasmine Hsu, J. Chase Kew, Marek Fiser, and Tsang-Wei Edward Lee. Long-range indoor navigation with PRM-RL. *IEEE Transactions on Robotics*, 36(4):1115–1134, 2020.

[2] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas A. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *IROS*, 2018.

[3] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. RL-CycleGAN: Reinforcement learning aware simulation-to-real. In *CVPR*, 2020.

[4] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *ECAL*, 1995.

[5] Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *ICML*, 2019.

[6] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018.

[7] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17:59:1–59:35, 2016.

[8] Zili Yi, Hao (Richard) Zhang, Ping Tan, and Minglun Gong. DualGAN: Unsupervised dual learning for image-to-image translation. In *ICCV*, 2017.

[9] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, Kun Zhang, and Dacheng Tao. Geometry-consistent generative adversarial networks for one-sided unsupervised domain mapping. In *CVPR*, 2019.

[10] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for Atari. In *ICLR*, 2020.

[11] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *CVPR*, 2018.

[12] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, 2012.

[13] Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. ROBEL: robotics benchmarks for learning with low-cost robots. In *CoRL*, 2019.

[14] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[16] Tinghui Zhou, Philipp Krähenbühl, Mathieu Aubry, Qixing Huang, and Alexei A. Efros. Learning dense correspondence via 3d-guided cycle consistency. *CoRR*, abs/1604.05383, 2016.

[17] Gregory J. Stein and Nicholas Roy. Genesis-rt: Generating synthetic images for training secondary real-world tasks. In *ICRA*, 2018.

[18] Tinghui Zhou, Philipp Krähenbühl, Mathieu Aubry, Qi-Xing Huang, and Alexei A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In *CVPR*, 2016.

[19] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017.

[20] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019.

[21] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017.

[22] Fereshteh Sadeghi and Sergey Levine. CAD2RL: real single-image flight without a single real image. In *RSS*, 2017.

[23] 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. *CoRR*, abs/1812.07252, 2018.

[24] Irina Higgins, Arka Pal, Andrei A. Rusu, Loïc Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: improving zero-shot transfer in reinforcement learning. In *ICML*, 2017.

[25] Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. In *ICML*, 2019.

[26] Gregory J. Stein and Nicholas Roy. Genesis-rt: Generating synthetic images for training secondary real-world tasks. In *ICRA*, 2018.

[27] Jingwei Zhang, Lei Tai, Peng Yun, Yufeng Xiong, Ming Liu, Joschka Boedecker, and Wolfram Burgard. VR-Goggles for robots: Real-to-sim domain adaptation for visual control. *IEEE Robotics and Automation Letters*, 4(2):1148–1155, 2019.

[28] Rae Jeong, Yusuf Aytar, David Khosid, Yuxiang Zhou, Jackie Kay, Thomas Lampe, Konstantinos Bousmalis, and Francesco Nori. Self-supervised sim-to-real adaptation for visual robotic manipulation. In *ICRA*, 2020.

[29] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *ICRA*, 2018.

[30] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *NIPS*, 2016.

[31] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[33] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: model-based offline policy optimization. In *NeurIPS*, 2020.

[34] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, 2019.

[35] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[37] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *RSS*, 2018.

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540), 2015.

[39] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *ICLR*, 2017.

[40] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *ICRA*, 2018.

[41] Melissa Mozifian, Juan Camilo Gamboa Higuera, David Meger, and Gregory Dudek. Learning domain randomization distributions for transfer of locomotion policies. *CoRR*, abs/1906.00410, 2019.

[42] Manuel Kaspar, Juan David Muñoz Osorio, and Jürgen Bock. Sim2real transfer for reinforcement learning without dynamics randomization. In *IROS*, 2020.

[43] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NIPS*, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The hyperparameters are provided in Sec. 1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]