CycleGAN for $sim2real$ Domain Adaptation

Anchit Gupta
Department of Computer Science
Stanford University
anchitg@stanford.edu

Jonathan Booher
Department of Computer Science
Stanford University
jaustinb@stanford.edu

Abstract

Bridging the reality gap is the main challenge in deploying policies learned using large scale reinforcement learning in the real world. In this paper we investigate the applicability of a novel approach proposed in [ZTX+18] for one shot transfer of robotic manipulation policies learned in simulation. Unlike traditional methods of increasing the visual fidelity in simulation or domain randomization we translate the real-world scenes to looks more like those in simulation. This decoupled approach to policy learning does not require retraining of the policy and is both flexible and efficient. We highlight some challenges and show positive results of this approach in simulation reinforcing it's applicability for actual $sim2real$ in future work. For code, please click here

1 Introduction

Following the success of Deep Q Learning (DQN) [MKS+13] in exploiting a huge number of environment interactions and neural networks to play Atari games various other methods like Deep Deterministic Policy Gradients (DDPG)[LHP+15] and Proximal Policy Optimization (PPO) [SWD+17] have been shown to work even for high dimensional continuous control robotic manipulation tasks [ZWM+18; FZZ+18]. Physics simulators like MuJoCo [TET12] were created that allow for relatively high accuracy robot simulation and policies are trained using multiple parallel copies of the simulated environment to maximize experience throughput. State of the art Deep Reinforcement Learning(DRL) algorithms [FZZ+18] for continuous control typically require millions of environment interactions to solve even mildly challenging manipulation tasks. This makes real robot learning using DRL unfeasible for all but the simplest of tasks.

However the perception and dynamics gap between reality and simulation still remains and poses challenges that are difficult for policies trained in sim to account for directly. $sim2real$ methods attempt to bridge this gap in various different ways. In this paper we focus on visual control tasks where the robot besides it's internal state only has access to a camera stream of the workspace and must reason about the 3D world from these images to complete the task. We seek to address the perceptual gap by using generative models to align the distribution of images in real and simulation. Hence allowing policies trained in simulated environments to generalise to the real domain.

Compared to other methods our pipeline decouples policy learning from transfer resulting in efficiency and flexibility in terms of using the same policy for different versions of the environment by just retraining the transfer network. To transfer to a new setting we typically need of the order of 100 images and no further fine-tuning of the policy network.

2 Related Work

Traditionally methods like pix2pix [IZZE16] have explored image-to-image translation or domain adaptation using supervised learning with datasets having paired images. But this is not widely

Preprint. Work in progress.

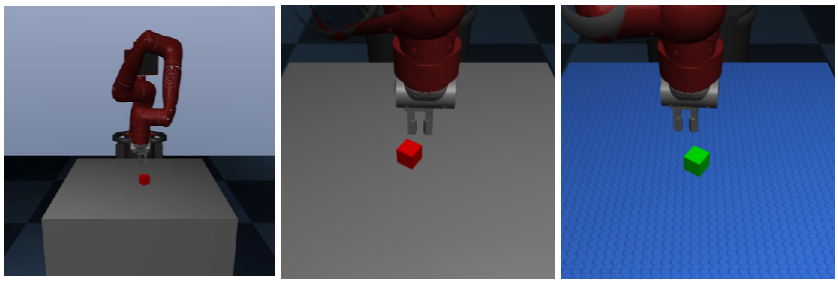

Figure 1: **Left:** SawyerLift environment **Center:** Agent view of "Simple" domain with flat textures **Right**: Agent view of "Complex" domain with textured surface

applicable due to the un-feasibility of having paired images for a wide range of domains. Unsupervised learning methods are well suited for such a task with unpaired images. One critical paper in this domain is CycleGAN [ZPIE17] which proposes using using two GANs and a cycle consistency loss to convert images between domains using just unpaired images. They show convincing results on many simple domains. CyCADA [HTP+17] extends the same with a semantic segmentation loss to improve performance on more complex distributions with multiple objects. A follow up paper Recycle-GAN [BMRS18] extends the above to videos by adding a temporal consistency term to the loss.

Domain randomization has been used to transfer object localisation networks trained in sim to the real world in [TFR+17]. In robotics [ZWM+18] have used the same to train visually robust policies and transfer them. Such techniques suffer from needing much longer to train due to the additional complexity that the policy has to learn to deal with and also needs a well designed randomization framework.

Our approach most closely resembles the recent VR-Goggles [ZTX+18] approach. They decouple policy learning from transfer and use GAN's to learn a transfer network to convert images from the real world to ones that look like simulation. Equipped with this domain adaptation network they are able to achieve one-shot sim2real transfer by running the policy on these transferred images. They test this on a simple indoor navigation task with a simulator that is already quite photo real hence reducing the complexity of the transfer task. In contrast we investigate this approach on a robotics manipulation task which is more complicated than navigation specially due to the the contact dynamics. These dynamics need to be inferred by the robot directly from images and hence small discrepancies can compound. Moreover the simulator we use is far from photo-real and hence the gap between sim and real is even larger.

3 Problem Statement

Due to the difficulty in setting up a real robot and calibrating the dynamics across simulation and real we first attempt to evaluate our method entirely in simulation. We consider the SawyerLift task in RoboSuite[FZZ+18] which involves teaching a robotic arm to lift a cube from pixel input 1. We have created two versions of the environment with differing color schemes and textures but the same underlying dynamics so that we can test our perceptual transfer method without any confounding factors.

We collect two types of data on each of the two versions of the environment. The first type is images from random movements of the robot to get a wide variety of views of the robot to hopefully learn a robust network to adapt between the two environments. The second type of data is generated from rollouts of our trained policy to get images of interactions with the cube. To get images on the "complex" environment, we add noise to the states gathered from rollouts. Note that in the future, it would be possible to use human demonstrations in the "complex" environment in order to gather images of object interaction in the "complex" domain.

Under this setting we expect to be able to learn a network mapping images from the source (unedited environment) to the target environment such that a policy trained entirely on the source task is able to work without any significant drop in performance on the target task by using the perceptual transfer

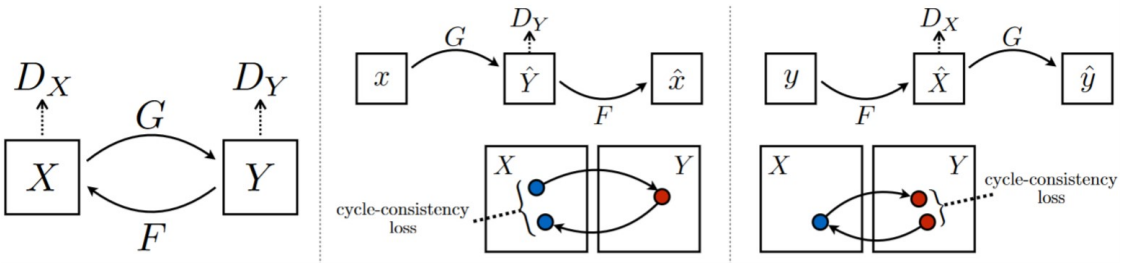

Figure 2: Cycle GAN Model schematic

network. Generated image quality can be evaluated qualitatively for relative cube positions/ radiations and presence of artefacts. Numerically this can be evaluated by the average rewards obtained by the policy when transferred via the learned domain adaptation network.

4 Technical Details

4.1 Policy Training

Policy gradient methods have been firmly established as the method of choice for continuous control problems like robotics. They operate by directly maximizing the expected sum of rewards $J(\theta)$ with respect to the parameters θ of the stochastic policy $\pi_\theta(a|s)$.But policy gradient estimates can have high variance (e.g. [DCH$^+$16]) and algorithms can be sensitive to the settings of their hyperparameters.

Several approaches have been proposed to make policy gradient algorithms more robust. Proximal Policy Optimization(PPO) is a robust policy gradient method that uses a soft trust region to deal with high variance monte carlo reward estimates. PPO has been shown in [FZZ$^+$18] to work really well in continuous control domains and hence is our method of choice. Specifically we use a state of the art distributed version Surreal-PPO [FZZ$^+$18] which supports massive parallelization across nodes for fast training. We train all our policies using 256 separate agents which are GPU(Tesla K80) accelerated for rendering and a learner with a P100 GPU. Some extra logging, eval and replay buffer nodes are used as well.

4.2 Domain Adaptation Network

CycleGAN (Fig 2) [ZPIE17] learns a mapping between two distributions X, Y via a pair of generators $G : X \rightarrow Y$ and $F : Y \rightarrow X$, a pair of discriminators D_X, D_Y which respectively distinguish between $X, F(Y)$ and $Y, G(X)$ are also trained in tandem. Besides just training a pair of GAN's the main novelty here lies in the cycle consistency loss which tries to force G, F to be inverses of each other i.e $G(F(Y)) \approx Y, F(G(X)) \approx X$. We use a simple L_1 distance to measure this as more complex formulations do not result in any experimental improvement. The overall loss function is

$$\min_{G,F,D_X,D_Y} L_{GAN}(G, D_X, X, Y)+L_{GAN}(F, D_Y, X, Y)+\lambda(E_X(||F(G(X))-X||_1)+E_Y(||G(F(Y))-Y||_1))$$

where λ is a tuneable hyperparameter. From our experiments, we set $\lambda = 10$. All 4 of the networks described above are trained together using gradient descent to optimise the above loss function.

We note that in the context of $sim2real$ for robotic control, images are sequential which results in small shifts of the images passed as input the generator. Often, these small differences in the images will result in large changes in the output of the generator (see Fig. 3) even when the cycle loss is small. To address this problem with vanilla CycleGAN, we abopt the $shift\ loss$ of [ZTX$^+$18] which augments the loss above to ensure similarity in approximately sequential frames.

$$E_{y,\ i,j\sim u(1,s-1)}[||F(y_{x\rightarrow i,y\rightarrow j}) - F(y_{x\rightarrow i,y\rightarrow j})||]$$

where the subscripts denote shifts of the image in pixels along the x and y axes, and s is the downsampling factor of the network. This loss encourages small shifts of the input image to result

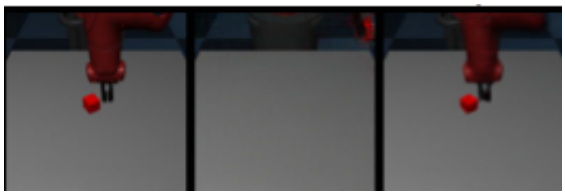

Figure 3: Shifting in generator ouptut in sequential frames

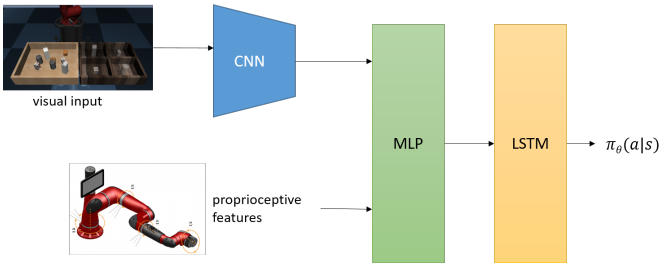

Figure 4: Policy network architecture for PPO

in equal shifts in the generated images. This allows for images generated sequentially to result in sequential generated images.

In this way, our full CycleGAN objective can be written as follows, where the s are additional hyperparameters to tune.

$$L_{GAN_X} + L_{GAN_Y} + \lambda_{cyc}\Big(L_{cyc_X} + L_{cyc_Y}\Big) + \lambda_{shift}\Big(L_{shift_X} + L_{shift_Y}\Big)$$

5 Experiments

We ran experiments for domain transfer on two versions of the above "lifting" task, one with simple, flat textures, and the other with a more complicated texturing (see Fig. 1). We take a trained policy π_θ that was trained on pixels (84X84 image) and robot arm position as input using Surreal [FZZ+18] on the "Simple domain". We use a hybrid CNN-LSTM architecture (Fig. 4) for the policy network which combines both the pixel input with the low dimensional robot joint data and feeds it into a LSTM module. The policy were trained using PPO(Proximal Policy Optimization). This trained policy achieves an average score of 340 on its source domain and a score of just 4.28 when used on the "Complex" domain. See Table 1 for a full summary of results.

5.1 Setup and Baseline

Using the trained policy and random exploration, we gather a dataset of 4000 images on each domain with a split of 1300 coming from policy rollouts and 2700 coming from random exploration. Images from policy rollouts on the "complex" domain were gathered by taking the low dimensional states from rollouts on the "simple" domain, adding random noise, setting the sim state, and rendering the complex domain. The CycleGAN is constructed using a generator architecture of a 6 block ResNet while the discriminator is a PatchGAN. The PatchGAN architecture classifies images based on "patches" which are sections of the image. This architecture has outperformed traditional ConvNet based discriminators when complex textures are involved. We set the scale of the shift loss to 0.003 and the cycles to 10.

Naive transfer of the policy to the textured environment failed to complete the task of lifting the object and received an average reward of 52.7 (grey gripper) and 4.28 (black gripper).

5.2 Effect of data imbalance

A first order experiment trained the CycleGAN using a balanced dataset of 4000 images on each domain. Training on this dataset showed, throughout most of training, a tendency to produce artefacts

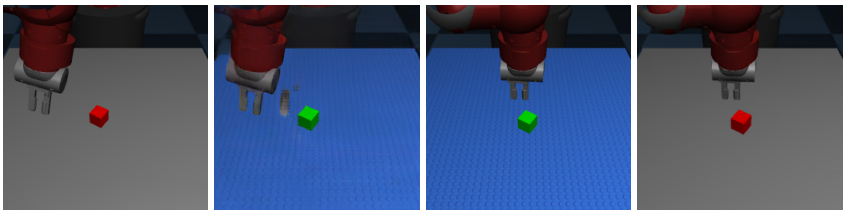

Figure 5: **Results from training on balanced dataset** From left: Real "simple domain", Generated "complex domain", Real "complex domain", Generated "simple domain"

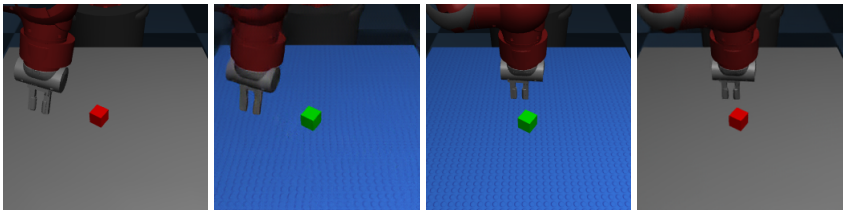

Figure 6: **Results from training on imbalanced dataset** From left: Real "simple domain", Generated "complex domain", Real "complex domain", Generated "simple domain"

with the arm in certain conformations as well as a tendency to shift and rotate the object which poses difficulties during transfer. Results at the end of training can be seen in Fig. 5. This model failed to transfer resulting in the policy being able to consistently "reach" the cube but failing to lift which resulted in an average return of 90.1.

Subsequent experiments focused on modulating the amount of data that was used for training on the "complex" domain to better approximate the challenges of $sim2real$ which create a large imbalance because training in simulation is cheap and multiple times faster than real-time. Initial experiments used a set of 4000 images on the "simple" domain and 400 on the "complex" domain which approximates the imbalance one would expect in $sim2real$. Results at the end of training the GAN can be seen in Fig. 6. Results on imbalanced data showed less generator artefacts as well as preserved the position and rotation of the cube. However, this model failed to transfer as well resulting in a failure to complete the task and a return of 112.4 which is an improvement over both previous attempts. Most of the errors produced by this model corresponded to the gripper disappearing in certain configurations of the arm which resulted in erratic policy action (Fig. 7).

Taking the idea of data imbalance to the extreme, further experiments were done using 4000 images on the "simple" domain and only 80 on the "complex" domain to push the limits of training using imbalanced data. These experiments will be detailed in the next section.

5.3 Changing gripper Color

To address the problem noted earlier of the disappearing gripper (Fig. 7), and to move the simulated environment closer to that of the real robot, we change the gripper color to a darker color (see the left

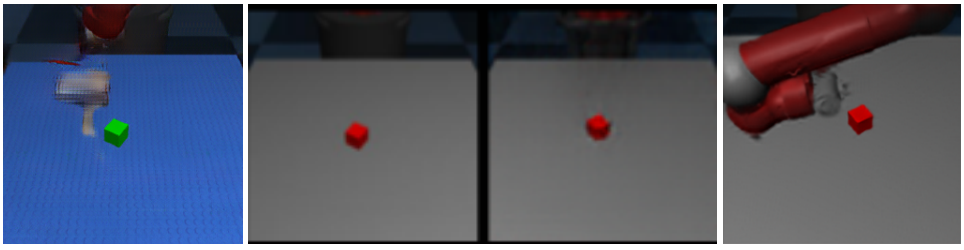

Figure 7: **Left** Example artefact created by the Generator **Center** Example of cube movement **Right** Example of the disappearing gripper

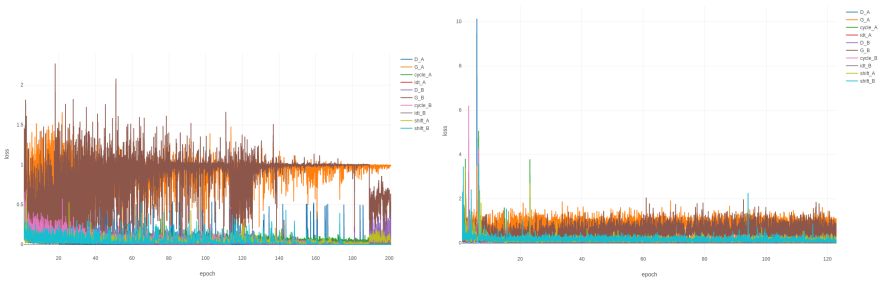

Figure 8: **Loss Plots left:** plots of losses when training on 84x84 images, it is clear that the model has diverged. **right:** plots of losses when training on 256x256 images, training is much more stable

part of Fig 9 for an example). We then train the CycleGAN to adapt using an imbalanced dataset of 4000 images on the "simple" and 80 on the "complex" domains.

Training using this dataset yielded positive results with the GAN able to bridge the perceptual gap and the transferred policy completed the task with 95% accuracy as well as an average reward of 320 and success rate of 95% which we classify as full transfer of the policy performance.

5.4 Effect of Image Resolution

Multiple experiments were run generating images of different resolutions, namely $84x84$ and $256x256$. The two sizes were motivated by the fact that the policy was trained using $84x84$ images and a common image size for benchmarks is $256x256$. For experiments with $256x256$ images, the environment generates full resolutions images which are fed through the generator and then downsampled to the correct size for the policy. The downsampling we hypothesised, would help with perceptual transfer as the operation of shrinking the image would naturally remove some of the imperfections and artefacts in the generated samples. Both resolutions allowed for successful policy transfer although the higher resolution GAN proved more consistent in its ability to transfer (95% vs. 40% of trials resulted in task completion) see Table 1 for a complete listing of rsults.

We note that transfer performance on the higher resolution images likely is more effective because of the noted regularisation effect of downsampling as well as the fact that training on larger images proved more stable whereas training on the smaller images required extensive parameter tuning and multiple runs to achieve a model that allowed for transfer. See Fig. 8 for plots of the losses that illustrate the difficulty of training on small images.

5.5 Transfer of Different Meshes

Our simulation contains two different meshes for its physical appearance, one the "visual" mesh approximates the way that the real robot appears, and the other "collision" mesh is a simpler mesh used to speed up calculation of collisions in simulation. These meshes differ in appearance Fig. 9. We sought to explore the possibility of using the CycleGAN to transfer between these two meshes. This helps to evaluate the possibility of using the CycleGAN to adapt between sim and real where the actual shapes of objects are different. To simplify the problem, we perform experiments adapting between to versions of the "simple" domain, one with each mesh visible.

Naive transfer of the policy to the environment with the collision mesh was able to accomplish the task, but with a low average return of 291.7 and a low success rate of 85%, the policy tended to have difficulty picking the cube up as well as dropping the cube once lifted. Using a trained CycleGAN between the two meshes resulted in a policy that was able to accomplish the task with an average reward of 305 with a success rate of 90%. The images produced by the CycleGAN (Fig 10) closely match that of the other domain in many cases; however, in some cases, it cannot fully capture the change in shape of the arm.

We ran experiments on both resolutions of images for transfer of mesh appearance and found that only training on higher resolution images resulted in successful policy transfer and qualitative results. Despite our best efforts at hyperparameter tuning, the best results at the low resolution can be seen in Fig. 10 and fell victim to extreme variations in robot position, even though the cycle loss was low

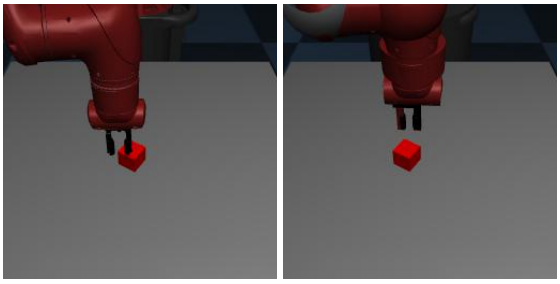

Figure 9: **Example Mesh visuals** From left: "visual" mesh, "collision" mesh. Note the change in shape of the arm as well as the change in color of the "elbow" joint. Changes in shape have proved difficult for CycleGAN in previous works [ZPIE17]

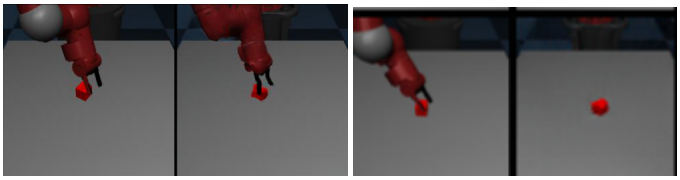

Figure 10: **Example Mesh Transfer** From left: Transfer of mesh visuals using 256x256 images, transfer of mesh visuals using 84x84 images.

indicating that an inverse transformation was learned, but the forward transformation is of low visual quality.

Baselines				
Experiment	Image Size	Gripper Color	Return	Success Rate
Baseline	84	Grey	365.9 ± 71.9	95%
Baseline	84	Black	340.2 ± 115.5	95%
Texture Transfer				
Experiment	Image Size	Gripper Color	Return	Success Rate
Naive Transfer	84	Grey	52.7 ± 34.7	0%
Naive Transfer	84	Back	4.28 ± 2.96	0%
Balanced Dataset	84	Grey	90.1 ± 9.86	0%
Imbalanced Dataset	84	Grey	112.4 ± 13.2	0%
Imbalanced Dataset	84	Black	155.8 ± 148.4	40%
Imbalanced Dataset	256	Black	$\mathbf{320.0 \pm 122.2}$	**95%**
Mesh Transfer				
Experiment	Image Size	Gripper Color	Return	Success Rate
Naive Transfer	84	Black	291.7 ± 105.4	85%
Imbalanced Dataset	84	Black	12.5 ± 6.43	0%
Imbalanced Dataset	256	Black	$\mathbf{305.4 \pm 113.0}$	**90%**

Table 1: **Results**

6 Conclusion

In this work we tackle the perceptual gap between sim and real using generative models. We show the applicability of the approach proposed by [ZTX+18] on a more complex robotic manipulation task. We achieve positive transfer results on the toy setup we have and also evaluate experimentally the effects of things like data imbalance, image resolution etc. which would help inform our decision when using this method on actual $sim2real$.

One obvious area of future work is to see if the same techniques can be used on the more difficult task of transferring between the real robot and simulation. Investigating whether more complex

manipulation tasks with more/varied objects like block stacking pick and place or nut assembly can also be solved using these methods is another area of future work. We might also need to further augment the GAN loss function to enforce object position/rotation consistency using things like semantic segmentation or edge detection.

Finding new ways to use demonstrations on the real robot to both improve quality of the CycleGAN and also try to improve policy transfer by more directly trying to account for the dynamics gap is also another avenue.

References

[BMRS18] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. *arXiv preprint arXiv:1808.05174*, 2018.

[DCH+16] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. *ICML*, 2016.

[FZZ+18] Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, 2018.

[HTP+17] 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. *CoRR*, abs/1711.03213, 2017.

[IZZE16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016.

[LHP+15] 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 e-prints*, page arXiv:1509.02971, September 2015.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[TET12] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Oct 2012.

[TFR+17] Joshua 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. *CoRR*, abs/1703.06907, 2017.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[ZTX+18] Jingwei Zhang, Lei Tai, Yufeng Xiong, Ming Liu, Joschka Boedecker, and Wolfram Burgard. VR goggles for robots: Real-to-sim domain adaptation for visual control. *CoRR*, abs/1802.00265, 2018.

[ZWM+18] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei A. Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, and Nicolas Heess. Reinforcement and imitation learning for diverse visuomotor skills. *CoRR*, abs/1802.09564, 2018.