TaskGAN: Combining Domain Loss and Task Loss for Domain Adaptation with GANs

Jonathan Heng [1] **Edward Johns** [1]

Abstract

Collecting large labelled datasets of real images can be highly time consuming and expensive. An alternative is to learn an image translation model which can be used to create large datasets of realistic looking images from easily obtainable and labelled simulated images. Specifically, we want to learn an image translation model $G : X \rightarrow Y$ that learns to map source domain images X (i.e. simulated data) to target domain images Y (i.e. real data). In this paper, we present a novel method, TaskGAN, for training generative adversarial networks to learn to generate a corresponding target domain image $G(X)$ from a source domain image X by minimizing the task loss of the generated target domain image. We also investigate the learning of the inverse mapping $F : Y \rightarrow X$ and find that it leads to better results. We show on a target reaching task in simulation that datasets created from a TaskGAN trained on 800 target and 22656 source domain images can be used to effectively train a neural network and achieve performances significantly better than a neural network trained only on 800 target domain images, with the inverse mapping approach achieving the best performance improvements of 55-59% across various target domains.

1. Introduction

In many modern machine learning algorithms for robotic tasks in the real world, large amounts of real image data with annotations or labels are required. However, such datasets can be highly time consuming and expensive to collect. On the other hand, annotations are easily available and data can be generated faster in a simulated environment. Hence, it would be ideal to be able to use simulated data to supplement the absence or lack of real world data. However, even while simulators are able to render increasingly realistic images, there still exists a difference between real data and simulated data. This difference is also known as the reality gap.

The reality gap is a classic transfer learning problem, where the source domain X is simulated data and the target domain Y is real world data. Transfer learning can be defined (Pan & Yang, 2010) as improving the learning of a target predictive function f_Y on a task T_Y in the target domain Y using the knowledge in the source domain X, where either the domains or the learning tasks are different. Here, we focus on the scenario where the domains are different, such as when transferring knowledge from a simulated environment to the real world. This subset of transfer learning problems is also commonly termed as domain adaptation.

Our approach to this problem is to first learn an image translation model of converting simulated images to corresponding realistic looking images, then creating a large dataset of realistic looking data with the labels of the simulated data, and finally training a neural network using the new dataset. The novelty in our work lies in the way our image translation model, TaskGAN, is trained. A detailed model of TaskGAN can be found in Figure 1. By utilizing a pre-trained task model f_Y in the target domain (i.e. real images), we can backpropagate the losses from the generated target domain image $G(X)$ to the generator G. The main motivation of using a pre-trained task model is because it contains semantic knowledge of the data in the target domain and will be able to guide the generator in reconstructing the semantic content from the input source image. An advantage our approach has over other image translation models that utilize a pixel-based loss (Bousmalis et al., 2016) or L1 loss (Shrivastava et al., 2016) for encouraging content reconstruction is that it does not assume visual similarity between source and target domains. This allows greater flexibility and ease when designing and creating the simulated source environment (e.g. the realistic qualities of the target domain need not be replicated in the simulated source domain). We term the learning of $G : X \rightarrow Y$ as the forward mapping approach.

After investigating the forward mapping approach, we discovered some drawbacks. Firstly, the task model f_Y is trained under the limitations of limited target domain data

[1]Imperial College London.

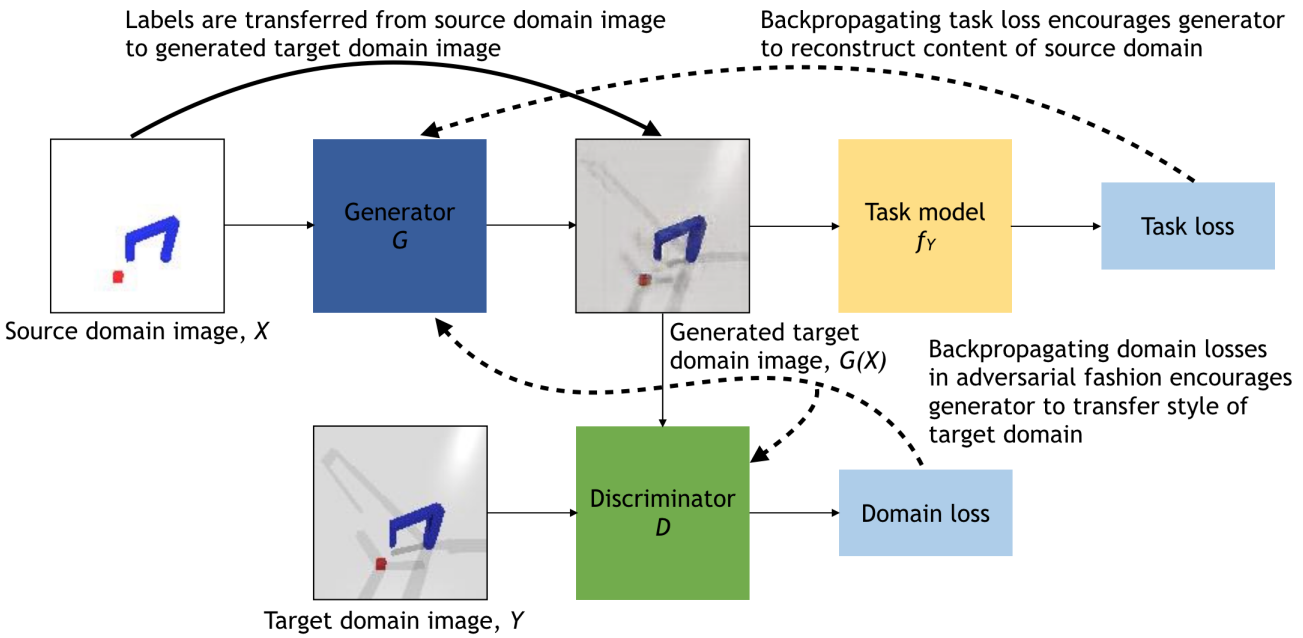


Figure 1. TaskGAN is an image translation model adapted from generative adversarial networks. Through adversarial training, the generator is encouraged to reconstruct an input source image in the style of the target domain. In addition, backpropagating task loss from the generated target domain image encourages the generator to reconstruct the content of the input source image. The figure above depicts the learning of a forward mapping (i.e. source $\rightarrow$ target) TaskGAN. The task is to predict the joint velocities that will move the robot arm towards the red cube. This task can be generalized to other regression tasks such as predicting joint angles and classification tasks.

(i.e. real images). While our experiments show that a weakly trained task model f_Y can improve the quality of generated images, the task model remains as a weak component that restrains the overall performance. A separate but related approach is to solve to the inverse problem of learning an image translation model that generates simulated looking images from real images, $F : X \rightarrow Y$. Instead of creating a large dataset of realistic images, we can train a task model with abundant simulated images and during test time, convert the realistic image to a simulated image. We also find that fine-tuning this model with the generated simulated images is helpful to performance as the model learns to ignore the noise created from the generator. We find the inverse mapping approach having a couple of advantages over the forward mapping approach. First of all, it is easier to train a task model f_X in the source domain (i.e. simulated images) since we assume that simulated images are easily available and annotated. Secondly, the inverse mapping tends to be easier to learn due to the nature of the target domain (i.e. real images) being more complex. The forward mapping is akin to a one-to-many mapping especially when we introduce noisy effects such as lighting and background changes. On the other hand, the choice of the source domain (i.e. simulated images) can be a simple and plain environment, leading to the inverse mapping being akin to a many-to-one mapping. Intuitively, a many-to-one mapping is much easier to learn and more robust.

An area of concern for the inverse mapping method is that the generator is trained on a small set of images (from the target domain in this case). Consequently, we may worry that the generator will fail to generalize well to unseen data. However, this does not seem to be an issue and we address it in Section 4.1.3. The steps to learning and utilizing both mappings for transfer learning are summarized in Figure 2. Overall, we find that the inverse mapping approach produces better results.

In this paper, we describe an approach to transfer learning which learns an image translation model as an intermediary step. Section 2 describes related works. Section 3 describes the overall methodology and details of the image translation model, TaskGAN, in greater detail. Section 4 covers two sets of experiments, a 3-degree-of-freedom robot arm control task in simulation and a robot arm state prediction task. Section 5 concludes the work and points out future directions.

2. Related work

Transfer learning for robot control: In (Tzeng et al., 2016), the authors proposed a weak pairwise constraint method in order to achieve automatic annotation. This method attempts to pair annotated synthetic (i.e. source domain) data with an unannotated real world counterpart (i.e. target domain) by finding a nearest neighbouring source image for each target image. In a different work (Rusu et al., 2016), the transfer learning problem is tackled by designing a neural network architecture that can utilize knowledge learnt in various domains. This architecture, termed progressive networks, retains the previous columns of networks learnt for different tasks, freezes the old weights, and adds a new network column for each new task. In (Zhang et al., 2016), the addition of simulated data to real world data for training leads to performance improvements for a planar reaching task.

The authors in (Bousmalis et al., 2017) proposes to first learn a model of converting simulated images to corresponding realistic looking images, then creating a large dataset of realistic looking data with the labels of the simulated data, and finally training a neural network using the new dataset. This work follows very closely to the forward mapping approach suggested in this paper but with key differences in the learning of the image translation model. More image translation models are described in the "Image translation with GANs" section under related work.

Domain adaptation and randomization: Some domain adaptation methodologies follow the idea of learning a set of domain invariant features. In other words, corresponding pairs of source and target domain images are reduced to a similar set of features. Works by (Tzeng et al., 2014; Rozantsev et al., 2016) minimizes a statistical distance such as the maximum mean discrepancy (MMD) between layers in neural networks.

Domain randomization is the concept of training a model on varied environments and data. The hypothesis is that if the variability in training data is sufficient, the model will be able to generalize well to unseen test data. In (Tobin et al., 2017), an object detector is trained entirely on simulated data with a wide range of randomly generated attributes including textures, shapes, and colours and achieved good accuracy when tested on real data. The work in (James et al., 2017) uses domain randomization to achieve an end-to-end model for a multi-stage robotic control task involving a robot arm grasping, moving, and dropping a cube from one area to another.

Generative adversarial networks (GANs): The paper (Goodfellow et al., 2014) that introduced GANs had an architecture that took Gaussian vectors as inputs and generated images from a specified domain. In general, a GAN architecture consists of a generator and a discriminator, and the generator aims to create outputs that can fool the discriminator. However, the original architecture is not sufficient for our desired application since we want to achieve a pairing effect where the generator can take an input source domain image and generate a paired target domain image. In other words, we want to transfer the style from the target domain and retain the content from the source domain.

Style transfer / Image translation: Image translation is an idea that has been explored at least since the work done in (Hertzmann et al., 2001). A more recent work used convolutional neural networks (Gatys et al., 2016) and minimized a mixed objective between content reconstruction and style transfer. However, these approaches were developed to transfer the style of a single image as opposed to our work which learns to transfer the style of a group of images.

Image translation with GANs: Adaptation of generative adversarial networks (GANs) for image translation have proven to be highly effective. The domain adversarial training of GANs encourages the transfer of style. Most works introduce modifications to incentivize the generator to create a pairing effect (i.e. to reconstruct the content from the input image) across domains. Examples include introducing a cycle consistency loss (Zhu et al., 2017), training the discriminator to differentiate between real source-target domain pairs $(X - Y)$ and fake source-generated target domain pairs $(X - G(X))$ (Isola et al., 2016), minimizing the L1 loss between the source domain image X and generated target domain image $G(X)$ (Shrivastava et al., 2016), and a content similarity loss (Bousmalis et al., 2016) that penalizes the difference between source and generated images for foreground pixels only.

3. Method

The overall methods are summarized in Figure 2. The main contribution of this paper is the TaskGAN model in step 2 and the majority of this section will focus on describing the details of this model. Using the forward mapping as reference, the aim is to learn a mapping from a source domain X to its corresponding image in the target domain Y. In addition to the architecture of a typical generative adversarial network, we propose the addition of a task model f_Y that is trained to perform a regression task such as predicting joint velocities and states of a robot arm. This task model can be generalized to any other task with labels such as a classification task. The intuition is that the task model f_Y contains knowledge of how the data should look like in the target domain Y when given a label from the source domain ϕ^X and backpropagating the loss from the generated target domain image $G(X)$ to the generator G will guide the generator to learn the mapping $G : X \rightarrow Y$.

	Forward mapping	Inverse mapping
Step 1	Learn a task model f_Y in the target domain *Y*.	Learn a task model f_X in the source domain *X*.
Step 2	Train a TaskGAN to learn the forward mapping *G : X -> Y*.	Train a TaskGAN to learn the inverse mapping *F : Y -> X*.
Step 3	Refine task model f_Y with generated target domain images *G(X)*.	Refine task model f_X with generated source domain images *F(Y)*.
Test time	Target domain data *Y* can be directly input to the task model f_Y to obtain results.	Target domain data *Y* needs to be converted to generated source domain data *F(Y)* before inputting to the task model f_X to obtain results.

Figure 2. Method summary

Firstly, given a target domain Y with images y_i and labels ϕ_i^y, we can pre-train a neural network for the task model $f_Y : Y \rightarrow \phi^Y$. We want the labels across domains to be transferrable in order for the task model f_Y to be helpful in guiding the generator in reconstructing the content when training the image translation model. In other words, for each x_i-y_i pair, the labels ϕ_i^x-ϕ_i^y should be identical. This will allow the source domain label ϕ_i^x to be used as the target domain label ϕ_i^y and a squared loss can be used as the task loss of the generated target domain image by computing $(f_Y(G(x_i)) - \phi_i^x)^2$.

Next, we train the image translation model, TaskGAN. Similar to a typical generative adversarial network, there is a discriminator and a generator. Here the discriminator network D_Y learns to differentiate between the real target domain data Y and the fake target domain data $G(X)$. The key difference is the training of the generator. In addition to creating images that will fool the discriminator, the generator also learns to create images that will minimize the task loss based on the source domain data label and the pre-trained task model f_Y.

Lastly, after learning a satisfactory image translation model, we fine-tune the pre-trained task model f_Y with a larger amount of newly generated target domain images $G(X)$.

3.1. Objective

The overall objective can be expressed as follows:

$$\begin{aligned} L_{TaskGAN}(G, D_Y) = \mathbb{E}_{y\sim Y}[D_Y(y)^2] \\ + \mathbb{E}_{x\sim X}[(1 - D_Y(G(x)))^2] \\ + \lambda_\phi \mathbb{E}_{x\sim X}[(f_Y(G(x)) - \phi^x)^2] \end{aligned} \tag{1}$$

where λ_ϕ corresponds to the weight of the task loss of the generated image $G(x)$. The weight used in implementation is 0.01. We find that larger values tends to more unstable training and quality of style transferred is reduced. Note that instead of using the typical log likelihood objective for generative adversarial networks, we use a least square loss. This follows the implementation details from (Zhu et al., 2017) where a least square loss was reported to perform more stably during training.

Through adversarial training, we aim to solve the following minimax game:

$$G^* = \arg\min_G \max_{D_Y} L_{TaskGAN}(G, D_Y) \tag{2}$$

$$\begin{aligned} L_G(G, D_Y, X, \phi^X) = \mathbb{E}_{x\sim X}[(1 - D_Y(G(x)))^2] \\ + \mathbb{E}_{x\sim X}[(f_Y(G(x)) - \phi^x)^2] \end{aligned} \tag{3}$$

$$\begin{aligned} L_{D_Y}(G, D_Y, X, Y) = \mathbb{E}_{y\sim Y}[(1 - D_Y(y))^2] \\ + \mathbb{E}_{x\sim X}[D_Y(G(x))^2] \end{aligned} \tag{4}$$

Given a minibatch of size N, equations 3 and 4 can be rewritten in the following manner:

$$\begin{aligned} L_G(G, D_Y, X, \phi^X) = \frac{1}{N}\sum_{i=1}^{N}[(1 - D_Y(G(x_i)))^2 \\ + (f_Y(G(x_i)) - \phi_i^x)^2] \end{aligned} \tag{5}$$

$$\begin{aligned} L_{D_Y}(G, D_Y, X, Y) = \frac{1}{N}\sum_{i,j=1}^{N}[(1 - D_Y(y_j))^2 \\ + (D_Y(G(x_i)))^2] \end{aligned} \tag{6}$$

3.2. Network architecture

The choice of discriminator and generator architectures in this paper is adapted from the following works (Johnson et al., 2016; Zhu et al., 2017). The naming convention follows those found in the appendix of (Zhu et al., 2017). We also replace the use of batch normalization with instance normalization as the work in (Ulyanov et al., 2016) reported better performances with the use of instance normalization.

Generator architecture: Let `c7s1-k` denote a 7x7Convolution-InstanceNorm-ReLU layer with *k* filters and stride 1. `dk` denotes a 3x3Convolution-InstanceNorm-ReLU layer with *k* filters and stride 2. `Rk`

denotes a residual block that contains a 3x3Convolution-InstanceNorm-ReLU-3x3Convolution-InstanceNorm with a skip connection and `k` filters in the convolutional layers. `uk` denotes a 3x3TransposedConvolution-InstanceNorm-ReLU layer with *k* filters and stride 2.

The generator network consists of:
`c7s1-32, d64, d128, R128, R128, R128, R128, R128, R128, u64, u32, c7s1-3`

Discriminator architecture: Let `C-sm-k-NR` denote a 4x4Convolution-InstanceNorm-LeakyReLU layer with *k* filters and stride *m*. Let `C-sm-k-R` denote a 4x4Convolution-LeakyReLU layer with *k* filters. Let `C-sm-k` denote a 4x4Convolution layer with *k* filters. The final layer is passed through a sigmoid activation function. We use a leak value of 0.2 for the leaky ReLUs.

The discriminator network consists of:
`C-s2-32-R, C-s2-64-NR, C-s2-128-NR, C-s1-256-NR, C-s1-1`

Task model architecture: Let `c-k` denote a 3x3Convolution-ReLU layer with *k* filters and stride 2. Let `fc` denote a fully connected layer with 256 output units. Another fully connected layer is placed at the end and the number of output units depends on the problem (e.g. if the problem is to predict the joint velocities for 3 joints, there will be 3 output units in the final fully connected layer).

The task model network consists of:
`c-16, c-32, c-64, c-128, c-128, fc`

In the forward mapping case, the availability of only a small amount of target domain data can be prohibitive in training an effective task model. We find that it can be helpful to use domain adaptation techniques such as a two column architecture inspired by the work in (Yosinski et al., 2014; Rozantsev et al., 2016). By treating neural networks as feature extractors and adding a loss that penalizes the Euclidean distance between the feature layers, it encourages the separate feature extractors for distinct domains to map pairs of images to the same feature space. The layer before the output layer is chosen as the feature layer in this implementation. We find improved performance when training with the L2 loss between feature layers and when the target domain feature extractor is initialized with the weights of a feature extractor trained in the source domain.

In the inverse mapping case, there are no need for such domain adaptation techniques since we assume that source data is easily available in large amounts.

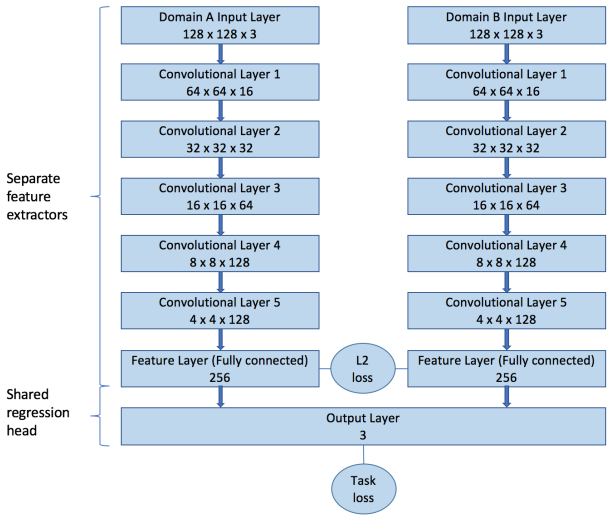


Figure 3. Task model using a 2 feature extractors architecture.

4. Experiments

4.1. Controlling a robot arm in a simulated environment

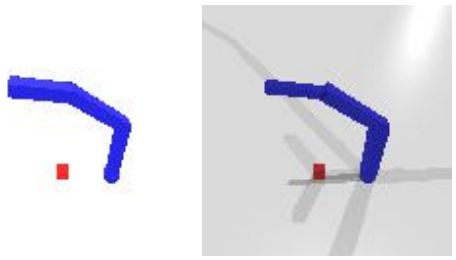


Figure 4. Images captured by the vision sensor in different V-REP scenes. Different domains can be created by adjusting the scenes. The image on the left represents the source domain and the the image on the right represents the target domain. The domain transformation is created by adding effects such as lighting, slight colour changes, and slight shape deformations to simulate the difference between simulated and real data.

4.1.1. Experiment set-up

Premise: The goal is to control the robot arm in a simulated environment to move towards a target object (e.g. a red cube). A custom dataset was created using V-REP. A source domain is treated as one where data is easily available, such as a simulated environment. A target domain is one where data is limited and expensive to obtain. We want to study the transfer of knowledge from a simulated environment (i.e. source domain) to a real environment (i.e. target domain). This can be simulated by creating

two scenes and an example can be found in Figure 4. The task is formulated as a regression problem of predicting the joint velocities given the image of the scene. We will train a task model as described in section 3.2 for the control of the robot arm.

Generating training data: The inverse kinematics module in V-REP allows us to create paths that move the tip of robot arm from a starting arm configuration to a position near the target object. This allows us to create images annotated with the joint angles. We convert the joint angles to joint velocities by taking the difference between the next state and the current state. We further normalize the joint velocities using the absolute norm and multiply them by a damping factor. The damping factor used is the distance between the tip and target. We find that using the damping factor leads to better performance as the arm will slow down and stop near the target object.

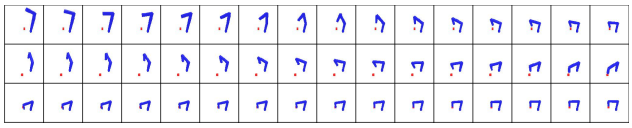


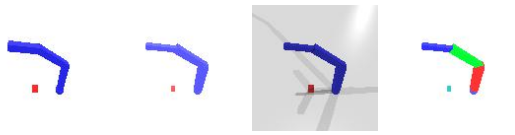
Figure 5. Examples of training images. Each row represents an episode of 16 steps starting from a random arm configuration and cube position. From left to right, the robot arm moves towards the cube.

Evaluation metrics: A test set consisting of 100 starting configurations is used to evaluate the performances of the models. Each episode is played over 100 time steps, where at each time step, vision sensor image is passed to the neural network and the predicted joint velocities is applied to the robot arm. The final distance mean and standard deviation is recorded. Success rate is also recorded. A distance threshold is used to decide if an episode is a success. If the final distance is below the threshold, the episode is considered successful. We use a threshold of 0.1m.

4.1.2. Forward (Source → Target) mapping

Training the task model f_Y: In the case of the forward mapping, the task model will be trained to perform in the target domain. We also limit the target datasets to a size of 800 images. For these datasets, we also make available the corresponding paired source datasets. The model is pre-trained on a source domain dataset of 22656 images and for 15000 steps. The target task model is further trained for 15000 steps with the small source-target paired dataset of size 800. For each training step, a batch size of 16 is used. The Adam optimizer with a learning rate of 0.001 is used.

Sample images of the various domains being tested on can be found in Figure 6.



(a) Source (b) Target A (c) Target B (d) Target C

Figure 6. Sample images from source and target domains.

Comparing against a typical GAN: We want to investigate whether adding task loss during training improves the quality of image translation. We do this by comparing samples of images generated from a typical GAN and a TaskGAN. The results are found in Figure 7. We compare the generated target domain samples $G(x)$ with the ground truth target domain samples y by superimposing the images. The TaskGAN method results in higher alignment quality, hence showing adding task loss during training has a positive effect on the quality of image translation.

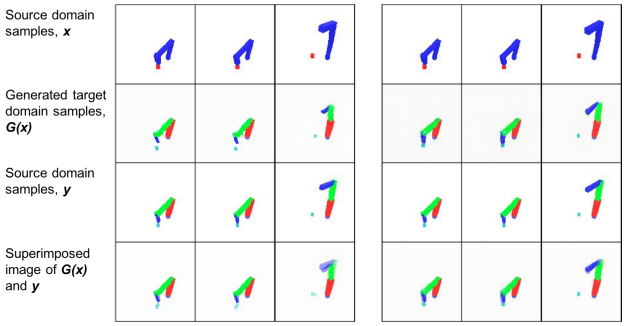


Figure 7. Generated samples from a TaskGAN are of higher quality than those generated from a typical GAN.

4.1.3. Inverse (Target → Source) mapping

A different approach to the problem is to learn the inverse mapping. This turns out to be an easier problem for a few reasons. Firstly, the task model for the inverse problem is in the source domain, where data is easily available. This allows us to use a well trained task model in the training of the TaskGAN. Secondly, we also find that the inverse mapping is easier to learn due to the nature of the target domain being more complicated. The forward (source-to-target) mapping is akin to a one-to-many mapping especially when we introduce noisy effects such as lighting and background changes. On the other hand, the choice of source domain can be a simple and plain environment, leading to the inverse (target-to-source) mapping being akin to a many-to-one mapping. Intuitively, a many-to-one mapping is much easier to learn and more robust.

We might be concerned about the generalizing ability of the inverse TaskGAN since the generator is trained on a much smaller set of target domain images. In a target domain with a wood background and light effects, the inverse TaskGAN trained with 800 target domain images shows good generalizing properties as seen in Figure 8. It is also worth noting that for this particular target domain, we were unsuccessful in learning the forward mapping even when using a well trained task model in the target domain.

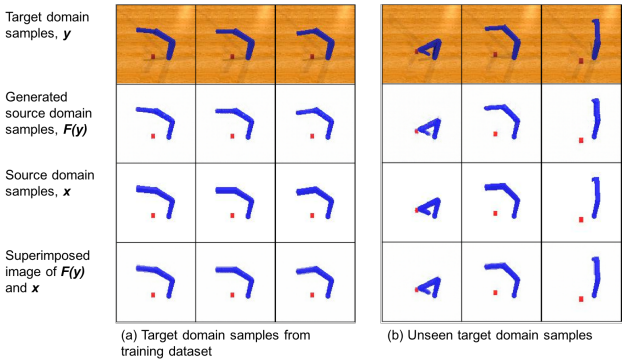


Figure 8. Inverse TaskGAN trained on a target domain dataset of 800 images generalizes well on unseen data and produces good quality image translations.

4.1.4. Results

After successful training of the image translation model, we fine-tune the task model with the translated images. This corresponds to step 3 as described in Figure 2. In the forward mapping case, we compare the performances of task models trained with a combination of the following datasets:

Benchmark: These are models trained with 800 images from respective test domains. This will serve as a baseline for comparing the effects of transfer learning using our method.

Source: Source domain dataset that contains 22656 images.

Paired: Small paired source-target domain dataset that contains 800 pairs of images.

Translated: Translated paired source-generated target domain dataset that contains 8000 pairs of images.

We train 5 task models ("Benchmark", "Source", "Source + Paired", "Source + Translated", "Source + Paired + Translated") for each target domain and compare the performances. The final results are summarized in a graph in Figure 9.

In general, the greatest performance gains are found when the task model is trained with the "Source + Paired + Translated" datasets. We find that just simply using the large amount of generated target domain data in the "Source + Translated" task model does not lead to as high a gain in performance. This is due to the generated target domain images being noisy versions of the true target domain images and the addition of the "Paired" dataset introduces a small number of true target domain images that helps the neural network learn to ignore the noise that are present in the generated target domain image. We also find that the overall performance gain in domain Target C is low and that is due to the generated target domain images being low in quality. The general flaws of doing the forward mapping include a weak task model for training a TaskGAN and a low quality of generated target domain images. These problems are alleviated when using the inverse mapping approach.

For the inverse mapping case, we train an "Inverse" task model. This model is initialized with weights from the pre-trained model in source domain. We further fine-tune the model with the generated source domain images $F(Y)$ created from the target domain dataset of 800 images. The results are summarized in Figure 9. We find general improved performance across all the target domains when compared against the forward mapping methods and also an overall higher quality of translated images. When comparing against the benchmarks, we find that task success rates increased by 55-59% across the various test domains.

4.2. Predicting the joint angles from real images of a robot arm

4.2.1. Experiment Set-up

Premise: The aim is to predict the joint angles of a Mico robot arm given an image of it. A corresponding simulated scene in V-REP is created to test the effects of performance when limiting the amount of real data and utilizing simulated data. Samples of the dataset can be found in Figure 10. The real Mico dataset contained 5399 images and are joint angles are annotated.

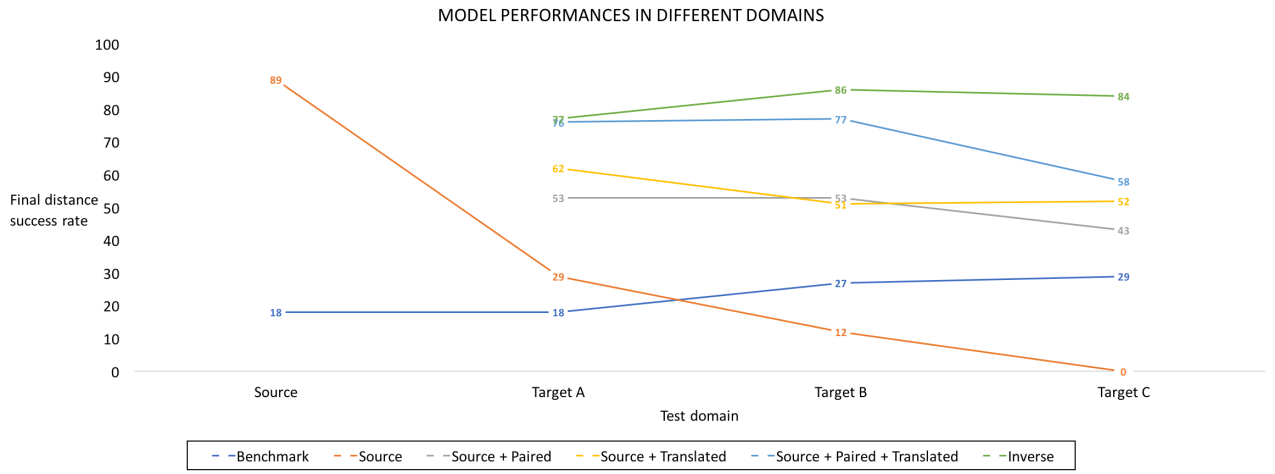


Figure 9. Test results in various domains under different training regimes.

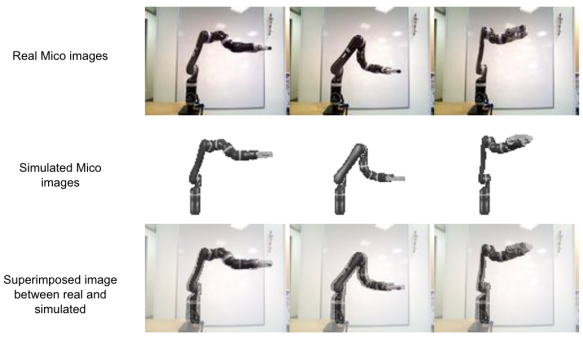


Figure 10. Images of the real Mico dataset and its corresponding simulated images created in V-REP. The superimposed images show the alignment between the two sets of images.

Modification to the problem: The original Mico dataset contains 6 joint angles for each image. However, due to the symmetrical and rotational property (i.e. when the joint is annotated at x degrees, it is visually identical when annotated at $x \pm 180$ degrees) of the last joint, which is the gripper of the arm, training a GAN to do the translation on this task is rather unstable. Modifying the problem to predict the other 5 joint angles lead to a much more stable performance.

Learning the inverse (Target $\rightarrow$ Source) mapping: In the previous section, we found that the inverse mapping is more effective and easier. Here we train inverse TaskGANs with varying amounts of real Mico images used during training and compare its effects. First, we train a task model f_Y in the simulated domain Y. This is trained with a dataset of simulated Mico images created using the corresponding V-REP scene. Note that the joint angles are randomly chosen and without knowledge of the real Mico dataset. Also, images with joints that are occluded are also not removed from the dataset.

4.2.2. Results

Quality of generated simulated Mico images: We train inverse TaskGANs with varying amount of real Mico images being used to teach the discriminator. A sample of a simulated Mico image created by these inverse TaskGANs on an unseen real Mico image can be found in Figure 11. We find that image clarity tends to improve as more real Mico images are being used to train the inverse TaskGAN.

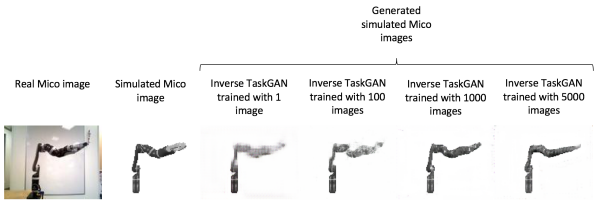


Figure 11. Generated simulated Mico images from inverse TaskGANs trained with varying amounts of real Mico images.

Data efficiency: Next, we test the data efficiency of the TaskGAN method. We study the extreme case when only a single target domain image is available (i.e. only 1 real Mico image is used for training). We train task models to predict joint angles when given a Mico image. The following models are tested:

Normal: A task model is trained on 1 real Mico image and tested on a real Mico image test set.

Inverse TaskGAN: A task model is pre-trained on 50000 simulated Mico images and an inverse TaskGAN is trained with this task model, 50000 simulated Mico images, and 1 real Mico image. During test time, the real Mico images from the test set are translated to simulated Mico images using the inverse TaskGAN and input into the task model for prediction.

Benchmark: A task model is trained on 50000 simulated Mico images and tested on a similar simulated Mico image test set.

	Normal	Inverse TaskGAN	Benchmark
Avg L2 loss	0.00848	0.00597	0.0021

Figure 12. L2 loss from predicting joint angles for various models.

From the results, we find that the "Inverse TaskGAN" approach gives an approximate reduction in L2 loss by 30% as compared to the "Normal" approach. There remains a significant gap to the ideal scenario as seen in the "Benchmark" case, where training data is abundant. However, under the constraints of highly limited target domain data, this approach remains viable for improving performances.

5. Conclusion

In this paper, we presented a method for transfer learning that utilizes the learning of an image translation model. Both forward and inverse mapping approaches are investigated and we find that the inverse approach has several advantages. The inverse approach reduces dependence on target domain images and relies on the concept of reducing complicated target domains to a simple source domain where data is easily available.

Another point worth noting is that the GANs trained in this paper are trained over a much shorter period of time compared to other similar works due to the lack of computational resources. It is still yet to be seen if extended training periods and larger datasets can lead to significantly better results. The robot control experiments were carried out entirely in simulated environments. More extensive experiments that shows the transfer of knowledge from a simulated environment to a real robot for a robotic control task will be highly interesting.

References

Bousmalis, Konstantinos, Silberman, Nathan, Dohan, David, Erhan, Dumitru, and Krishnan, Dilip. Unsupervised pixel-level domain adaptation with generative adversarial networks. *arXiv preprint arXiv:1612.05424*, 2016.

Bousmalis, Konstantinos, Irpan, Alex, Wohlhart, Paul, Bai, Yunfei, Kelcey, Matthew, Kalakrishnan, Mrinal, Downs, Laura, Ibarz, Julian, Pastor, Peter, Konolige, Kurt, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv:1709.07857*, 2017.

Gatys, Leon A, Ecker, Alexander S, and Bethge, Matthias. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2414–2423, 2016.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Hertzmann, Aaron, Jacobs, Charles E, Oliver, Nuria, Curless, Brian, and Salesin, David H. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 327–340. ACM, 2001.

Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

James, Stephen, Davison, Andrew J, and Johns, Edward. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *arXiv preprint arXiv:1707.02267*, 2017.

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

Pan, Sinno Jialin and Yang, Qiang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Rozantsev, Artem, Salzmann, Mathieu, and Fua, Pascal. Beyond sharing weights for deep domain adaptation. *arXiv preprint arXiv:1603.06432*, 2016.

Rusu, Andrei A, Vecerik, Matej, Rothörl, Thomas, Heess, Nicolas, Pascanu, Razvan, and Hadsell, Raia. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

Shrivastava, Ashish, Pfister, Tomas, Tuzel, Oncel, Susskind, Josh, Wang, Wenda, and Webb, Russ. Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828*, 2016.

Tobin, Josh, Fong, Rachel, Ray, Alex, Schneider, Jonas, Zaremba, Wojciech, and Abbeel, Pieter. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017.

Tzeng, Eric, Hoffman, Judy, Zhang, Ning, Saenko, Kate, and Darrell, Trevor. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

Tzeng, Eric, Devin, Coline, Hoffman, Judy, Finn, Chelsea, Abbeel, Pieter, Levine, Sergey, Saenko, Kate, and Darrell, Trevor. Adapting deep visuomotor representations with weak pairwise constraints. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

Ulyanov, Dmitry, Vedaldi, Andrea, and Lempitsky, Victor. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, and Lipson, Hod. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

Zhang, Fangyi, Leitner, Jürgen, Upcroft, Ben, and Corke, Peter. Vision-based reaching using modular deep networks: from simulation to the real world. *arXiv preprint arXiv:1610.06781*, 2016.

Zhu, Jun-Yan, Park, Taesung, Isola, Phillip, and Efros, Alexei A. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.