Sim-Real Joint Reinforcement Transfer for 3D Indoor Navigation

Fengda Zhu[†] Linchao Zhu[§] Yi Yang[§‡]
[†]UTS-SUSTech Joint Research Centre, Southern University of Science and Technology
[§]CAI, University of Technology Sydney [‡]Baidu Research
{zhufengdaaa,zhulinchao7}@gmail.com Yi.Yang@uts.edu.au

Abstract

There has been an increasing interest in 3D indoor navigation, where a robot in an environment moves to a target according to an instruction. To deploy a robot for navigation in the physical world, lots of training data is required to learn an effective policy. It is quite labour intensive to obtain sufficient real environment data for robots training while synthetic data is much easier to construct by rendering. Though it is promising to utilize the synthetic environments to facilitate navigation training in the real world, real environment are heterogeneous from synthetic environment in two aspects. First, the visual representations of the two environments have significant variances. Second, the house plans of the two environments are rather different. Therefore, two types of information, i.e.*, visual representation and policy behavior, need to be adapted in the reinforcement model. The learning procedure of visual representation and that of policy behavior are presumably reciprocal. We propose to jointly adapt visual representation and policy behavior to leverage the mutual impacts of environment and policy. Specifically, our method employs an adversarial feature adaptation model for visual representation transfer and a policy mimic strategy for policy behavior imitation. The experimental results show that our method outperforms the baseline by 21.73% without any additional human annotations.*

1. Introduction

Autonomous indoor navigation is a problem in which the robot navigates to a target according to an instruction within a building such as house, office and yard. This task will benefit many applications where a robot takes over human being's job, such as house cleaning, package delivery and patrolling. Solving indoor navigation in a 3D environment is the basis of these mobile robotic applications in the real-world scenario.

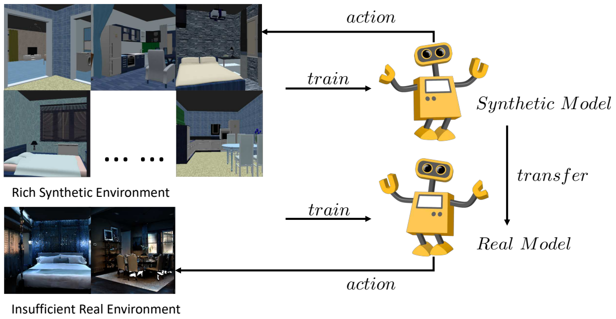


Figure 1. This figure illustrates the general framework about transferring knowledge from synthetic model (top) to real model (down). In testing, the two models receive images and predict actions (turn left, turn right or forward) to navigate in environments.

Earlier works adopt imitation learning methods including behavior cloning [3] and DAGGER [30] for 3D navigation tasks. These approaches train a robot to emulate an expert, such as the shortest path in indoor navigation. These approaches fail into over-optimization, which suppress a larger set of close-optimal solution. Deep reinforcement learning approaches based on actor-critic, *e.g.*, A3C [25] and UNREAL [20] are widely used in recent researches [27, 44, 40, 34]. Advantage of these end-to-end approaches is that it discretizes the agent and state space [48] and explores explicit map representations for planning [24].

There has been impressive progress on deep reinforcement learning (RL) for many tasks, such as Atari video games, GO [26, 35], robot control [10], self-driving [33] and navigation [40]. However, it is difficult to apply reinforcement learning to the physical environment since sampling a large number of episodes for training a robot is time-consuming and even impossible. Thus, recent researches focus on the RL model to learn policy in a simulated environment rather than in the physical world to resolve the problem [40, 34]. There are two types of environments that can be used to train a robot. The first one is rendered synthetic environment. A represent work is SUNCG [36],

Part of this work was done when Yi Yang was visiting Baidu Research during his Professional Experience Program.

Corresponding author: Yi Yang.

which produces a 3D voxel representation from a single-view depth map observation. The second one is reconstructed environment such as Matterport3D [5], which consists of real images captured with a Matterport camera.

One problem of only using the Matterport3D dataset for training, however, is the lack of diversity of scenes. the SUNCG dataset [36] consists of over 45,622 synthetic indoor 3D scenes with customizable layout and texture, while Matterport3D [5] contains only 90 houses. Lack of training houses and scenes can significantly hamper performance because of overfitting. Models trained on the SUNCG dataset, on the contrary, is more capable of generalizing to unseen scenes.

In this paper we propose a joint framework, namely Joint Reinforcement Transfer (JRT), to resolve the problem of lacking training data in the real environment. Our premise is that synthetic training data is much easier and cheaper to obtain than real data. As shown in Figure 1, our algorithm integrates adversarial feature adaption and policy mimic into a joint framework. The joint learning of the adversarial feature adaption and policy mimic makes them mutually beneficial and reciprocal. In this way, the feature adaption and policy mimic are tightly correlated. The adversarial feature adaptation not only generates better representation well fits real environment, but is also more suitable for policy imitation.

It is intuitive to transfer visual feature from synthetic domain to real domain, so that we can directly adopt knowledge learned from synthetic environment without changing the policy function. This kind of method, also called Domain Adaptation, is widely applied in numerous tasks [22, 32, 43, 23, 9, 38]. Inspired by the idea of Adversarial Discriminative Domain Adaptation, we use the adversarial loss to supervise our transfer training process. Compared to image translation methods like CycleGan [45], which directly translates the visual image with GAN [13, 15, 4, 47], our approach drops the redundant steps that generates target images and extracting feature for target domain. Therefore, our model is pimplier in framework and less parameter in model so that be easy to converge.

In addition, the policy could be tuned under the supervision of teacher trained on synthetic environment. Our motivation is two-fold. First, the layout of real environment is not exactly the same as the synthetic environment. As shown in Figure 2, houses of real environment has more rooms and much more connections between rooms, which means more complicated structure. Second, some knowledge learned in synthetic scene, such as finding doors and bypassing obstacles is also helpful in real environment navigation. Thus, we propose our Joint Reinforcement Transfer(JRT) to integrate the two training stages together.

Our experimental results show that this joint method outperforms baseline with finetune by 21.73% without any additional human annotations. Also, By qualitatively visualizing the results of the two adaptation methods, we shows the two parts are mutually enhanced to benefit real-world navigation.

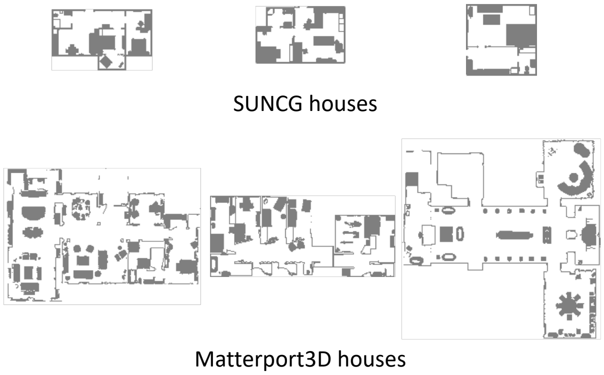


Figure 2. This figure shows the different house plans of the SUNCG dataset and Matterport3D dataset. Houses in SUNCG are smaller and simpler in layout. Meanwhile, houses in Matterport3D have more rooms and are more complicated in house plan.

2. Related Work

The proposed work is related to transferring reinforce policy trained on virtual environment to real environment. In this section, we briefly review several methods on 3D Navigation, Indoor Environment, Domain Adaptation and Policy Transfer with high relation to our topic.

3D Indoor Environment There has been a rising interest in indoor reinforce environment. House3D [40] is a manually created large scale environment. AI2-THOR [21] is an interactable synthetic indoor environment. Agent can interactive with some interactable objects such as open a drawer or pick up a statue. CHALET [42] is another interactable synthetic indoor environment with larger interactive action space. Recent works tend to focus on simulated environment based on real imagery. However, the scale of these datasets are quite small compared with virtual datasets. Active Vision dataset [42] consists of dense scans of 16 different houses. And Matterport3D [5] is a larger, multi-layer environment. Minos is a cross domain environment which consist both House3D and Matterport3D. MINOS provides similar setting of both environments, which is convenient for our environmental transferring experiment.

Domain Adaptation The problem of domain adaptation has been widely studied and arises in different visual application scenarios such as image classification [22, 32], object detection [43] and action recognition [11]. Prior work such as [19, 14] used Maximum Mean Discrepancy(MMD) [29] loss to minimize the difference between the source and target feature distributions. MMD computes the difference of data distributions in the two domains. Parameter adaptation, another kind of early method used in [43, 9, 17], adapt the classifier like SVM trained on the source domain.

In contrast, Recent works introduce adversarial approach into domain adaptation. Generative Adversarial Network(GAN) [13], lets generator G and discriminator D compete against each other, is widely applied because of its powerful ability for learning and generalizing from data distribution. Gradient Reversal [12] directly optimizes the mapping by reversing the gradient of discriminator. To go one step further, Adversarial Discriminative Domain Adaptation [38] uses a target generative model optimized by adversarial loss to learn source feature distribution. This method can better model the difference in low level features. There are several work [16, 41] focus on Synthetic-to-Real visual image translation and a benchmark called VisDA [28] has been proposed recently.

Policy Transfer Some works have explicitly studied action policy transferring in reinforcement learning senario in different way. Hinton *et al*. [18] proposes the method of Network Distillation, using student model to learn the output distribution of teacher model. Policy Distillation [31] employs this method to transfer knowledge between two environment based on DQN algorithm. Target-driven Visual Navigation [48] is a model for better generalize to new goals rather than new environmental scene. Semantic Target Driven Navigation [27] learns policy only based on detection and segmentation result, so that model can be easily transferred to unseen environments.

3. Model

3.1. Baseline Setup

We demonstrate our approach of reinforce model transfer based on room goal navigation task. Before our approach, we will first present our general model under the framework of standard reinforcement setting.

According to partially observable Markov decision process (POMDP), we can formulate our problem as (S, A, O, P, R). The agent starts from a initial state $s \in S$, which is the pose of the agent, a position direction pair. The action $a \in A$ is a discrete set of predefined actions. Observation space is the union of multi-modal sensory input space such as RGB image, depth image and force $O = \{O_{rgb}, O_{depth}, O_{force}...\}$. Probabilistic state-action transition function is represented as $p(s_{i+1} \mid s_i, a_i)$. Reward $R(s, a)$ is a function related to the Euclidean distance to the goal and normalized time [34].

LSTM A3C LSTM A3C is a general model of asynchronous advantage actor-critic algorithm. Compared to Feedforward A3C whose policy only based on the current observation, LSTM A3C introduce temporal experience to achieve better performance.

LSTM A3C starts with a convolution network used as visual embedding module

$$f = CNN(O)$$

f stands for visual feature of current observation. The visual feature sequence from the step 0 to current step t is written as $f_{seq} = \{f_0, f_1, ..., f_t\}$.

Also we have a goal g to indicate which room agent is required to go. g is encoded into a semantic feature vector as g_{emb} by a embedding layer:

$$g_{emb} = G_{emb}(g)$$

Then visual feature of each step and goal embedding are concatenated to fed to (Long-Short Term Memory) LSTM units for temporal encoding

$$h_t = LSTM([f_t, g_{emb}], h_{t-1})$$

where f_t is the visual feature of current step and h_t is the LSTM output vector, encoding the historical information from step 0 to t. Note that we simplify the propagation of memory cells for convenience.

Follow the architecture of actor-critic, we have a policy module $a = \pi(a|h_t)$ to predict the distribution of ongoing actions and value $v = V(h_t)$ to predict the value of this state. Loss functions of A3C baseline is defined as

$$L_{A3C} = -log\pi(a|h_t; \theta)(R_t - V(h_t; \theta)) - H(\pi(a|h_t; \theta))$$

where term $H(\pi(s_t; \theta))$ represents the entropy of policy. We maximize this term to encourage exploration.

UNREAL UNREAL is an advanced version of A3C with several unsupervised auxiliary tasks providing wider training signals without additional training data. This improvement is general so that it can augment many reinforce tasks including navigation in this paper. We will prove that our approach can also be applied to UNREAL model in experiment and achieve the state-of-the-art on our setting.

Re-Formulation In the following sections, we will propose two method, Feature Adaptation and Policy Transfer. For simplify notation, we re-define the model as two parts, which consist of a mapping function M and a policy function P.

Mapping function M is fed with observation(only RGB images in our experiment) $x \sim X$ and output visual embedding $f \sim F$. We define X as RGB image distribution and F as feature distribution. $X_s, X_r \subseteq X$ represent image distribution of synthetic domain and real domain respectively. Similarly, we use $F_s, F_r \subseteq F$ to represent synthetic visual feature and real visual feature.

Policy function P takes f as input and predict discrete action with $softmax$ activation. Our representation of reinforce model is written as

$$\pi(a|s; \theta) = softmax(P(M(o; \theta_M); \theta_P))$$

We use L_{policy} to represent the united form of actor-critic loss for both A3C and UNREAL.

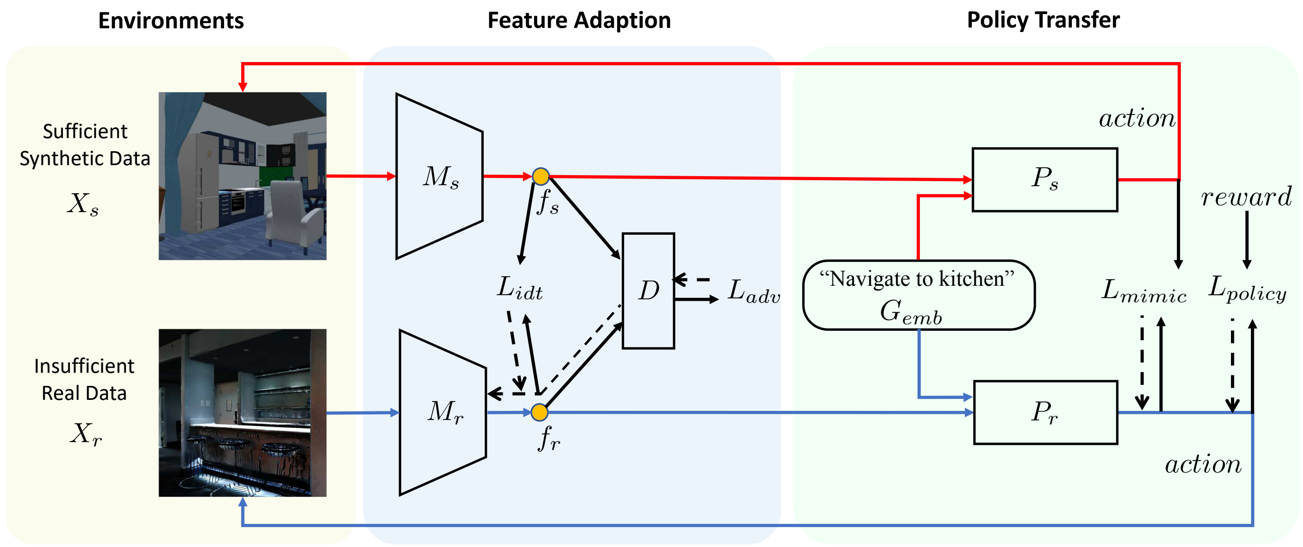


Figure 3. An overview of our Joint Reinforcement Transfer (JRT). Our model contains two RL models: synthetic model with M_s and P_s and real model with M_r and P_r. We use red line and blue line to represent the testing procedure of the two models. Black arrows stand for forward and dotted lines stand for backward in the training procedure. f_s and f_r are feature embeddings of synthetic image X_s and real image X_r, respectively. For feature adaptation, we apply adversarial loss L_{adv} on top of discriminator D. Additionally, we have identity loss L_idt to regularize the learned semantic embedding. In policy transfer, we have the mimic loss L_{mimic} trained with policy loss L_{policy}.

3.2. Adversarial Feature Adaptation

For adapting model trained from simulated environment to real environment, one intuitive idea is mapping visual observation to a latent space with same distribution. Thus the following policy network can be easily applied to real environment since its input distribution will not change significantly.

Motivated by this, we consider adversarial learning method to adapt the mapping function for real images. Note that we use A3C to demonstrate our approach for simplicity. Following the notation of [38], we assume X_s as image drawn from a distribution of synthetic image domain $p_{syn}(x, y)$ and Y_s as its label. Similarly, we have real image X_r and label Y_r from $p_{real}(x, y)$. We mark the mapping function for synthetic images as M_s. Thus we have

$$f_s = M_s(x_s)$$

Here f_s also follows a synthetic feature distribution represented as $f_s \sim F_s$. Our goal is to train a real mapping function M_r for visual embedding $f_r \sim F_r$, where f_s and f_r belongs to the same distribution.

$$fr = M_r(s_r)$$

$$f_s, f_r \in S, Fr = Fs$$

To build our adversarial learning procedure, we need a binary discriminator D to distinguish whether a feature f is mapped from X_s or X_r. Label l_k equal to 1 is when f is mapped from X_s and equal to 0 otherwise. Thus the classifier D is optimized by cross entropy loss below

$$\begin{aligned} L_{cls}(X_s, X_r) = & - \mathbb{E}_{x_s \sim X_s} \log(D(M_s(x_s))) \\ & - \mathbb{E}_{x_r \sim X_r} \log(1 - D(M_r(x_r))) \end{aligned}$$

We design our adversarial loss to optimize M_r so that M_r and C can compete each other

$$L_{adv} = \mathbb{E}_{x_r \sim X_r} \log(D(M_r(x_r)))$$

The optimization of M_r, which equal to maximizing $\mathbb{E}_{x_r \sim X_r} log(D(M_r(x_r)))$, will force the distribution of F_r getting closer to F_s, so that discriminator D is more difficult to distinguish F_s and F_r. By alternatively optimizing M_r and D, The visual embedding f_s and f_r will finally belong to same distribution.

However, $M_s(x_s)$ and $M_r(x_r)$ can still be mapped to different semantic vector in latent space. It is due to adversarial method above does not ensure f_s to have the same semantic representation as f_r. Inspired by the technique of [37], we used identity mapping loss to regularize the mapping function M_r to be an approximate function of identity mapping when input is a synthetic image x_s. We use L2 loss because our regularization is applied on feature space. This approach suppose that mapping function M_r has good generalization ability and synthetic image shares exact the same semantic space as real image. Thus we propose our identity mapping loss:

$$L_{idt} = \|M_s(X_s) - M_r(X_s)\|_2$$

By introducing L2 weight norm for both M_r and D respectively, our full objective of feature adaption is written as:

$$L_{total} = L_{adv} + L_{cls} + \lambda_1 L_{idt} + \lambda_2 L_{norm}$$

Although L_{adv} and L_{cls} are in same magnitude, we need to balance regularization terms by λ_1 and λ_2. How performance influenced by loss weight are fully discussed in experiment section.

We use unpaired images from synthetic environment and real environment to train our model. As shown in Figure 3, our gradient are only computed from adversarial loss L_{adv} and identity loss L_{idt}. Instead of reinforcement training procedure, we train model following the framework of ADDA [38]. Parameter of synthetic mapping function M_s are fixed while M_r and discriminator D is updated per step. We sample batches of images every training step, compute gradient from two losses and then update M_r and D.

In testing, M_r is connected with policy function P_r. Without Policy Transfer method, we can just simply copy the parameter from P_s to P_r and achieve a significant performance improvement from the adaptation of M_r.

3.3. Policy Mimic

Previously we have a powerful original policy network trained on large scale synthetic data and a mapping function to adapt visual embedding to real environment. However, due to the huge gap between synthetic environment and real environment, adapting visual embedding only is not enough for reinforce model transfer.

Therefore we introduce an auxiliary approach to transfer our reinforce policy. We find it is necessary to combine transferred policy with knowledge gained from real environment. Since our task is to predict next action by classification, we introduce Policy Distillation method [31], to transfer knowledge leaned from synthetic environment.

Our baseline trained on synthetic environment acts as a teacher model T. A student model learns knowledge from T is written as S. Both model is fed by real environment data. And the action probability p predicted by T is served as soft label to train S, which is also called mimic. Student model S is trained with a log-likelihood loss to predict the same action distribution:

$$p = softmax(P_s(M_s(x_r)))$$

$$\begin{aligned} L_{mimic} = &- \mathbb{E}_{x_r \sim X_r}[p \cdot log(softmax(P_r(f_r)))) \\ &- (1-p) \cdot (1 - log(softmax(P_r(f_r))))] \end{aligned}$$

Different from [31], which student model only supervised by $a_{best} = argmax(q)$ our student model S learns the full distribution of p from teacher T. This loss function enables student learn more knowledge such radios of very small probabilities from soft labels, as demonstrated in [18].

We optimize L_{mimic} together with L_{policy} in real environment. By balancing the weights between reinforce loss and policy transfer loss, our full objective of Policy Transfer is defined as

$$L_{total} = L_{policy} + \lambda L_{mimic}$$

Note that loss weight λ can be different when L_{mimic} is trained together with UNREAL [20] since UNREAL have several auxiliary tasks which make the magnitude of L_{policy} slightly larger.

Unlike Feature Adaptation, we follow the reinforcement learning framework of [25], training model in real environment only. As shown in Figure 3, synthetic model and real model take the same input from real environment X_r. M_r is pretrained by adaptation method last section. Parameters of M_s, P_s and M_r is fixed and only P_r is tuned by mimic loss L_{mimic} and policy loss L_{policy}.

In testing, as the blue line shown in Figure 3, P_r is connected at the bottom of M_r. Tuning policy function P_r can get another significant performance improvement.

4. Experiments

4.1. Datasets and Baseline Setup

Our setting is mainly based on MINOS dataset [34], for MINOS provide unified interfaces for both synthetic and real environments. Same configuration of scenes, multimodal sensory inputs and action space facilitate implementation.

Synthetic Environment SUNCG [36] is a large scale environment consists of over 45,622 synthetic indoor 3D scenes. It renders image observation with 3D occupancy and semantic labels for a scene from a single depth map. Following the setting of MINOS, we manually select a subset of 360 single-floor houses fit for room navigation. A large range of houses in SUNCG, however, are not appropriate for our task. For some scenes, rooms are not connected with each other and other scenes are not even houses. SUNCG scenes are split into 207/76/77 for training, validation and testing respectively.

Real Environment Matterport3D [5] is a multi-layer real-like environment with 90 scenes. We follow the same splitting strategy as original dataset. It has 61 rooms for training and 18 rooms for testing. We sample 10 episode for each test house for total 180 episode as our test set.

Baselines We consider two algorithms, A3C and UNREAL, which trained on SUNCG and Matterport3D from scratch as our baseline models. For each baseline model, we adopt

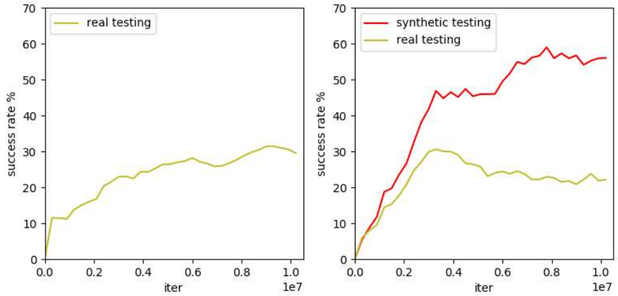


Figure 4. The left figure shows the testing results of real baseline on real environment. The right figure shows the testing results of synthetic baseline on synthetic and real environment respectively.

RMSProp solver with 4 asynchronous threads. Agent is trained for 13.2M total steps, corresponding to 1 day of experience on a Quadro P5000 GPU device.

There are three widely used types of goals for indoor navigation: PointGoal, ObjectGoal and RoomGoal. PointGoal instructs agent to go to a precise relative point, represented as (x, y). ObjectGoal makes agent to find a object in house. RoomGoal, as the most complex task, requires agent to find a specified room house. RoomGoal is quite difficult not only because texture of furnitures are in diverse color but also difference of its shapes and positions can make distinct observations. Moreover, to navigate efficiently enough, agents have to learn how rooms are connected, such as a bedroom is always connected with a bathroom and a kitchen is often connected with a living room. As a result, we adopt RoomGoal as goal type to prove our idea.

We test our baseline models on the sampled test episodes. Usually, performance of a navigation model is evaluated by success rate [2] report as percentage. Recent paper [1] proposes a new measurement called Success weighted by (normalized inverse) Path Length(SPL). This metric requires model to navigate to goal as possible when taking the optimal path. We evaluate our models on both metrics.

To show our baseline models are sufficiently trained, we test the success rate of our baselines for every 30K iterations. Figure 4 indicates both baselines converged after 8×10^6 iterations. However, synthetic baseline perform badly when tested on real environment and its performance continually drop after 8×10^6 iterations. This performance drop shows there is domain gap between synthetic and real environments.

4.2. Adversarial Feature Adaption Experiments

For unpaired adversarial training, we sample 10,000 RGB images from synthetic and real environment respectively. We adopt Adam optimizer with initial learning rate = 0.0001, β_1 = 0.9, β_2 = 0.999. Each training batch with size 64 consists of 32 synthetic images and 32 real images. This training process is easy to converge so that we test all the model at 1000 iterations.

method	% success rate	% SPL
sim baseline	24.13% ± 2.12%	22.46% ± 1.52%
real baseline	32.67% ± 0.73%	27.60% ± 2.33%
sim+FT	36.80% ± 2.64%	27.95% ± 1.18%
sim+FA	**56.27% ± 2.94%**	38.74% ± 0.85%
sim+PM	47.64% ± 2.25%	33.15% ± 2.37%
sim+FA+PM	**58.53% ± 2.96%**	**42.25% ± 2.23%**

Table 1. Success rate for different baselines and feature adaptation results. Here FT stands for finetune, FA means adversarial feature adaptation and PM means policy mimic.

Note that we use initialize the parameter of M_r with M_s because M_s content some semantic encoding. Without a good starting point, model is easily collapse and its performance could be even lower than baseline.

Comparison against baselines In Table 1, we compare three different baselines with our feature adaptation method. Among three baselines the best performance is achieved by fine-tuning on real environment after training on reinforcement synthetic environment. This baseline gets highest success rate because it is trained on both environment so that get more information. However, our method outperforms the best baseline greatly by 19.47%, even though baseline with finetune is trained with more information.

Note that baseline with fine-tune access more information than feature adaptation since fine-tune on real environment will not only get the image input but also get the reward supervision. Even though std is large since our scale of testing data is small, huge increasing of mean value showing feature adaptation method works on reinforcement policy transfer.

Analysis of the identity loss In Table 2, we compare the influence of different identity loss weight to adaption performance. We find that identity loss plays a important role in adversarial training. Without identity loss, which means its loss weight equals 0, performance will drop by 7.74%.

When identity weight goes even larger, mapping function for real environment M_r tend to act much more like M_s. It makes feature distribution F_r far away with F_s, which may led policy function P_r make wrong decision. When identity weight continue increasing, Performance of the whole model will drop and eventually be equal to synthetic baseline. Thus we adopt the hyper-parameter identity loss weight = 0.0005 in the following experiments.

4.3. Policy Mimic Experiments

Following the Policy Mimic training described in the Section 3.3, we only update policy function P_r in real environment.

idt ablation	% success rate
idt=0	$48.53\% \pm 2.93\%$
idt=$5 * e^{-6}$	$54.00\% \pm 2.23\%$
idt=$5 * e^{-4}$	$\mathbf{56.27\% \pm 3.94\%}$
idt=$5 * e^{-2}$	$53.07\% \pm 1.55\%$

Table 2. Ablation study: success rate for different weight of identity loss

method	% success rate
UNREAL baseline	$24.13\% \pm 2.12\%$
UNREAL baseline+FA	$56.27\% \pm 2.94\%$
UNREAL baseline+FA+PM	$\mathbf{58.50\% \pm 2.96\%}$
A3C baseline	$16.00\% \pm 2.02\%$
A3C baseline+FA	$34.80\% \pm 0.88\%$
A3C baseline+FA+PM	$\mathbf{50.00\% \pm 1.88\%}$

Table 3. Success rate for different baselines, feature adaptation(FA) and policy mimic(PM) results

mimic ablation	% success rate
UNREAL baseline	$24.13\% \pm 2.12\%$
UNREAL baseline+FT	$36.80\% \pm 2.64\%$
mimic weight=0	$36.00\% \pm 3.55\%$
mimic weight=0.1	$\mathbf{58.53\% \pm 2.96\%}$
mimic weight=0.2	$34.53\% \pm 1.95\%$
mimic weight=0.5	$22.80\% \pm 1.85\%$

Table 4. Ablation study: success rate for different weight for mimic loss. Here FT means baseline with finetune.

The environmental configuration and model hyperparameter is same as baseline when trained in real environment. We use M_r trained in Adversarial Feature Adaptation method and initialize P_r by P_s. Similar to feature adaptation, if we just random initialize our policy function, the result could be much lower. To prove our policy mimic method is general, we test our method on both A3C model and UNREAL model.

Comparison against baselines Table 3 summarizes our results and compares our policy mimic method with feature adaptation only and baseline.

UNREAL baseline is higher than A3C baseline for 8% since the auxiliary task provides additional training signals. However, after feature adaptation, the performance gap is increased to nearly 22%. A reasonable explanation is additional training signals are mainly beneficial to policy network. Thus feature transfer can be a greater help for UNREAL model compared to A3C model.

Finally, it is the policy mimic method that reduces the performance gap. Even though the teacher model for policy mimic of A3C is A3C baseline trained on synthetic environment, model can still absorb useful knowledge learned from synthetic data.

Analysis of the mimic loss Table 4 reports our ablation experiment upon mimic loss. Compared with ablation experiment for weight of identity loss, weight of mimic loss seems to be more sensitive. Performance will drop rapidly if mimic weight be slight deviate the optimal value. When mimic loss weight is 0, it is equivalent to finetuning model on real environment. Compared to finetune baseline above, since mapping function M_r has been transferred, the performance is slightly higher.

The reason why mimic loss is rather important is that the policy function can overfit easily in the real environmental without rich training data. Thus imitating behavior of model trained on synthetic model can solve this difficulty.

4.4. Visualization

In this section, we evaluate our methods from qualitative aspects.

We run different models on unseen real environments in the test set. Meanwhile, we record agents navigating trajectories, RGB input sequences and some intermediate results. By comparing these results, we testify that our methods are able to optimize intermediate procedure and therefore, improve final performance.

Visual Attention We now attempt to analysis how adversarial feature adaptation influence mapping function.

We plot sequences of heatmap for last convolution layer combining with RGB inputs. Figure 5 compares three different models with similar RGB images input.

It is obvious that baseline trained on synthetic data is not able to embedding real environment data. The model can not focus on specific targets. On the contrary, its attention is scattered on the whole map. Finally, it can not go to target room but miss the door by keep turning right.

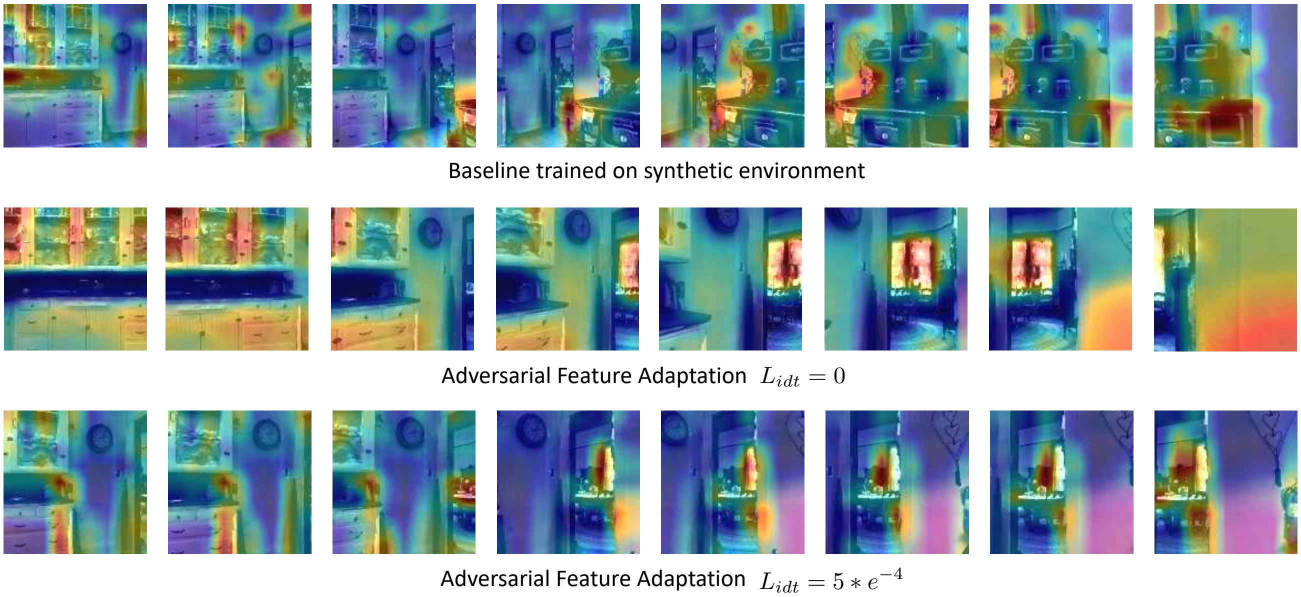
After adversarial training even though without identity loss, model learns to focus on some targets like cabinet, wall and window. However, these targets have minor effect on navigation. It means model can learn how to distinguish objects by adversarial adaptation. By paying too much attention on the wall at right, this model misses to cross the door either.

With identity loss augmenting adversarial training, model not only learns to recognize objects, but also know the semantic difference between targets. We find that this model pay more attention on door like edges and be able to find the real door.

Policy Behavior Based on model after adversarial adaption, we now analysis the policy behavior in some complex cases. Figure 6 shows a scene where model need to cross a door not being able to distinguish from RGB information easily.

Model without policy mimic does not have a stable navigation behavior. It repeatedly turns left or right in a random way. It seems that model is overfit on a local minimum, which prohibit its exploration.

In contrast, model with policy mimic gains a more stable navigation. It keeps turning following one direction to

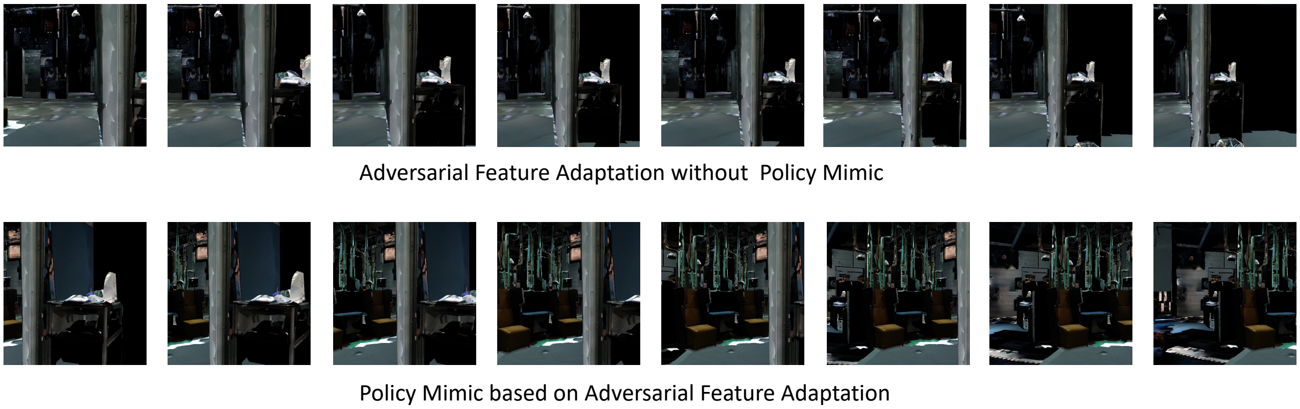


Baseline trained on synthetic environment

Adversarial Feature Adaptation $L_{idt} = 0$

Adversarial Feature Adaptation $L_{idt} = 5 * e^{-4}$

Figure 5. Heatmap for the last convolution in mapping function. Red and yellow regions are places with large values where network mainly focus on. We can capture the relationship between scene transitions from left to right and attention on heatmaps.



Adversarial Feature Adaptation without Policy Mimic

Policy Mimic based on Adversarial Feature Adaptation

Figure 6. RGB image sequences from left to right indicates the models' trajectories in testing.

explore scene of whole room.

This knowledge is learned from large scale synthetic data, where model is able to be trained among diverse scenes to avoid overfitting. The knowledge is suitable for navigation in real environment, however, which model can hardly learn from because of few training data.

5. Conclusion

To relieve the problem of lacking real training data, we propose JRT to discover the knowledge learned from synthetic reinforcement environment to facilitate the training process in real environment. Specifically, we integrate adversarial feature adaptation and policy network into a joint network. We demonstrated the effectiveness of the framework with extensive experiments. We also conducted ablation studies of the identity loss and mimic loss to show its superiority. Finally, our methods outperform baselines in both qualitative and quantitative ways without any additional human annotations.

In the future, we will validate the transfer ability on other tasks, *e.g.*, Embodied Question Answering [8] and exploit temporal information for feature adaptation [39, 46, 7] and feature analysis [6].

Also, policy mimic can be further improved by considering episode-wise optimization. In addition, it is meaningful to investigate how to transfer policy in vision and language navigation tasks.

Acknowledgments. We thank AWS Cloud Credits for Research for partly supporting this research.

References

[1] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 6

[2] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018. 6

[3] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 2009. 1

[4] J. Cao, Y. Guo, Q. Wu, C. Shen, J. Huang, and M. Tan. Adversarial learning with local coordinate coding. In *International Conference on Machine Learning*, pages 707–715, 2018. 2

[5] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 2, 5

[6] X. Chang and Y. Yang. Semisupervised feature analysis by mining correlations among multiple tasks. *IEEE transactions on neural networks and learning systems*, 28(10):2294–2305, 2017. 8

[7] X. Chang, Y.-L. Yu, Y. Yang, and E. P. Xing. Semantic pooling for complex event analysis in untrimmed videos. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1617–1632, 2017. 8

[8] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 5, page 6, 2018. 8

[9] L. Duan, I. W. Tsang, D. Xu, and S. J. Maybank. Domain transfer svm for video concept detection. 2009. 2, 3

[10] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016. 1

[11] N. Faraji Davar, T. de Campos, D. Windridge, J. Kittler, and W. Christmas. Domain adaptation in the context of sport video action recognition. In *Domain Adaptation Workshop, in conjunction with NIPS*, 2011. 2

[12] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 3

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2, 3

[14] A. Gretton, A. J. Smola, J. Huang, M. Schmittfull, K. M. Borgwardt, and B. Schölkopf. Covariate shift by kernel mean matching. 2009. 2

[15] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017. 2

[16] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla. Understanding real world indoor scenes with synthetic data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4077–4085, 2016. 3

[17] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[18] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3, 5

[19] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007. 2

[20] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations (ICLR)*, 2016. 1, 5

[21] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 2

[22] M. Long, J. Wang, G. Ding, J. Sun, and S. Y. Philip. Transfer feature learning with joint distribution adaptation. In *2013 IEEE International Conference on Computer Vision*, pages 2200–2207. IEEE, 2013. 2

[23] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[24] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016. 1

[25] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning(ICML)*, 2016. 1, 5

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 1

[27] A. Mousavian, A. Toshev, M. Fiser, J. Kosecka, and J. Davidson. Visual representations for semantic target driven navigation. In *Proceedings of european conference on computer vision*, 2018. 1, 3

[28] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924*, 2017. 3

[29] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence. Covariate shift and local learning by distribution matching, 2008. 2

[30] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011. 1

[31] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015. 3, 5

[32] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 2

[33] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017. 1

[34] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*, 2017. 1, 3, 5

[35] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017. 1

[36] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 190–198. IEEE, 2017. 1, 2, 5

[37] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016. 4

[38] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 4, 2017. 2, 3, 4, 5

[39] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018. 8

[40] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian. Building generalizable agents with a realistic and rich 3d environment. In *Proceedings of european conference on computer vision*, 2018. 1, 2

[41] J. Xu, D. Vázquez, A. M. López, J. Marín, and D. Ponsa. Learning a part-based pedestrian detector in a virtual world. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2121–2131, 2014. 3

[42] C. Yan, D. Misra, A. Bennnett, A. Walsman, Y. Bisk, and Y. Artzi. Chalet: Cornell house agent learning environment. *arXiv preprint arXiv:1801.07357*, 2018. 2

[43] J. Yang, R. Yan, and A. G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 188–197. ACM, 2007. 2, 3

[44] H. Yu, X. Lian, H. Zhang, and W. Xu. Guided feature transformation (gft): A neural language grounding module for embodied agents. In *Conference on Robot Learning (CoRL)*, 2018. 1

[45] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the ieee international conference on computer vision*, 2017. 2

[46] L. Zhu, Z. Xu, Y. Yang, and A. G. Hauptmann. Uncovering the temporal context for video question answering. *International Journal of Computer Vision*, 124(3):409–421, 2017. 8

[47] M. Zhu, P. Pan, W. Chen, and Y. Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[48] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 2017. 1, 3