LEARNING CATEGORY-LEVEL GENERALIZABLE OBJECT MANIPULATION POLICY VIA GENERATIVE ADVERSARIAL SELF-IMITATION LEARNING FROM DEMONSTRATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generalizable object manipulation skills are critical for intelligent and multi-functional robots to work in real-world complex scenes. Despite the recent progress in reinforcement learning, it is still very challenging to learn a generalizable manipulation policy that can handle a category of geometrically diverse articulated objects. In this work, we tackle this category-level object manipulation policy learning problem via imitation learning in a task-agnostic manner, where we assume no handcrafted dense rewards but only a terminal reward. Given this novel and challenging generalizable policy learning problem, we identify several key issues that can fail the previous imitation learning algorithms and hinder the generalization to unseen instances. We then propose several general but critical techniques, including generative adversarial self-imitation learning from demonstrations, progressive growing of discriminator, and instance-balancing for expert buffer, that accurately pinpoints and tackles these issues and can benefit category-level manipulation policy learning regardless of the tasks. Our experiments on ManiSkill benchmarks demonstrate a remarkable improvement on all tasks and our ablation studies further validate the contribution of each proposed technique.

1 INTRODUCTION

Learning human-level object manipulation skills is highly demanding for robots, which can automate repetitive works in our real-world complex scenes and lead to revolutionary applications, *e.g.* home robots, that may impose huge impacts on human society. A crucial and valuable feature of human manipulation skills is the remarkable generalization ability – for certain tasks (*e.g.* open the door), we can successfully manipulate different kinds of object instances (that contain a door), regardless of the large variations in their geometry and topology. With the emerging deep reinforcement learning (RL) and imitation learning (IL) techniques, we have recently observed great progress in generalizable robot learning on basic tasks that only involve rigid objects with simple manipulation skills, *e.g.* grasping (Levine et al., 2015; Mahler et al., 2017), planar pushing (Yu et al., 2016), pick and place (Zeng et al., 2020a), etc. However, learning generalizable policy for more complex tasks that involve articulated objects, *e.g.* open drawers and doors, are still highly under-explored.

Recently, to facilitate the research and benchmark the progress in learning generalizable object manipulation skills on articulated objects, Mu et al. (2021b) initiate SAPIEN open-source manipulation skill (ManiSkill) challenge. The challenge introduces four tasks with each aimed at manipulating one category of articulated objects using robots with either a single arm or dual arms. Based on SAPIEN environment and PartNet-mobilty dataset (Xiang et al., 2020), the benchmark split a category of objects into non-overlapped training and testing instances and the learned policy is expected to work on both seen instances as well as novel object instances at test time. As shown in Mu et al. (2021b), learning such category-level generalizable policy is very challenging for RL-based approaches. Naively adopting state-of-the-art RL methods on category-level generalizable policy learning achieves almost zero success rate. To ease the learning, ManiSkill provides many successful demonstrations on training object instances and thus enables learning from demonstrations.

In this work, we thus tackle the problem of learning category-level generalizable object manipulation policy via imitation learning in a task-agnostic manner. Although, for each individual task,

conducting reward engineering that incorporates rich task and object category priors into a handcrafted dense reward do help, we aim to find a more general and universal approach that assumes no handcrafted rewards but only a terminal one, which will benefit all kinds of category-level manipulation tasks and is complimentary to task-specific reward design.

With the sufficient amount of demonstrations, we couple the popular IL method, Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon, 2016), with prevailing model-free off-policy RL algorithm, Soft Actor-Critic (SAC) (Haarnoja et al., 2018) and achieve non-zero success rates on both training and test instances. However, this baseline is far from being satisfactory and not tailored for category-level generalizable policy learning.

Through in-depth investigation, we identify several key problems that hinder the generalizable policy learning: 1) to handle many training instances altogether, the policy network is very hard to get ramp up, which imposes a severe challenge to GAIL, whose reward provided by discriminator can easily diminish to zero, and fail the imitation; 2) the demonstrations of different object instances may be from different strategies, which is difficult for a single policy network to imitate, given its limited capacity and intrinsic continuity in neural networks; 3) even the policy learning is successful on some training instances, the learned policy can still be highly biased such that it can only handle a certain type of instances and doesn't generalize to novel instances.

To mitigate the aforementioned issues, we propose several important extensions to the baseline. Our technical contribution includes: propose to progressively grow the discriminator of GAIL to mitigate issue 1); propose Generative Adversarial Self-Imitation Learning from Demonstrations to mitigate issue 2); and propose category-level instance balancing expert buffer to mitigate issue 3).

Our proposed method achieves remarkable improvements on success rates, outperforming the GAIL+SAC baseline by **13%** and **18%** averaged across four tasks on training and validation sets, respectively. On the "no external annotation" track of ManiSkill Challenge 2021[1] that allows interaction, imitation, dense reward but no further annotations, our method ranks first place when further coupling with dense rewards.

2 Related Work

2.1 Learning Generalizable Manipulation Skills

Generalization is essential for future robot applications. The robots are expected to be functional in a variety of real-world environments. For simple motion tasks like pushing and lifting, the variation in objects seldom constraints the performance of algorithms. However, for dealing with articulated objects whose each part has unique dynamics and function, generalization becomes challenging. Previous works focus to identify key parts or extract features of articulations as representations (Mittal et al., 2021; Arduengo et al., 2021; Devin et al., 2018) to enable generalized manipulation on different instances. These methods are based on visual information including key location identification, pose estimation, or pretrained attention models. There are also control-based methods using model prediction and generative planning methods (Abbatematteo et al., 2019; Jain & Niekum, 2020) to achieve robust and adaptive control on both seen and novel objects.

2.2 Imitation Learning from Demonstrations

Imitation learning (IL) techniques aim to mimic expert behaviors in a certain task. When expert demonstrations of the tasks are given, a task-specific reward function becomes optional. The ordinary way to perform imitation learning including behavior cloning (Bain & Sammut, 1995) and inverse reinforcement learning (Ng et al., 2000; Russell, 1998). In recent years, a new IL method has emerged, namely, generative adversarial imitation learning (GAIL) (Ho & Ermon, 2016) which has achieved state-of-the-art performance on numerous complex robotics tasks (Merel et al., 2017; Wang et al., 2017). This method incorporates RL into the GAN framework. The generator which is the policy of the RL part generates demonstrations in the environment, and the discriminator is used to discriminate the generated and expert demonstrations, such that the policy of RL converges to the expert policy. To improve the sample efficiency, there have been numerous studies leveraging off-policy RL algorithms rather than original on-policy RL algorithms for policy generation (Kostrikov et al., 2018). In addition, GASIL (Guo et al., 2018) combines the idea of self-imitation (Oh et al., 2018)

[1] https://sapien.ucsd.edu/challenges/maniskill2021/

with GAIL to realize imitation learning without demonstration data. Moreover, SILfD (Pshikhachev et al., 2021) uses expert demonstration data to help self-imitation learning.

However, using imitation learning to train an agent that utilizes expert demonstrations effectively and generalizes to different unseen objects still remains challenging.

2.3 3D Articulated Object Manipulation

3D articulated object manipulation is at the nexus of computer vision and robotics and has been an attractive topic in research. Previous works utilize the learning methods to get the vision knowledge (such as articulated part configurations, link poses, joint parameters, and state information) of the environment from the 3D input in order to build a bridge between vision and robotics and help the downstream robotic tasks (Wang et al., 2019; Li et al., 2020; Zeng et al., 2020b; Mu et al., 2021a). Other works like Abbatematteo et al. (2019) estimate kinematic structures to predict the shape and kinematic model of an object from depth sensor data. On the other hand, there have also been impressive works about various robotic planning, trajectory optimization, and control methods on articulated object manipulation (Peterson et al., 2000; Chitta et al., 2010) like door-opening. These methods typically assume perfectly known geometry with objects pre-fixed to the robot. Recent works have demonstrated successful systems which leverage the vision learning methods and the robotic planning methods to manipulate 3D articulated objects with the learned visual knowledge (Gadre et al., 2021; Harada et al., 2019; Schmid et al., 2008). These works typically start with the vision methods such as pose estimation, object detection, and part segmentation to get the visual knowledge of the environment and compute a motion trajectory. However, this explicit standardized visual knowledge may be insufficient for the downstream robotic tasks if the diverse articulated objects have complex topological and geometric variations, such as different numbers and shapes of doors and drawers on different shapes of cabinets.

3 Problem Formulation

Our problem concerns the policy learning for robots to manipulate a variety of articulated objects from one object category for a specific task, e.g. opening different types of drawers. The tasks and the environments come from the ManiSkill Challenges (Mu et al., 2021b). The goal is specified by instance mask and part mask of the point cloud obtained from multiple depth cameras in the environment, for example, the target drawer and its corresponding handle are labeled in the opening drawer task, and the task is considered successful when the target drawer is opened to a certain extent and remain static. Assuming the robot state is always known, our state S_t thus comprises a colored point cloud with per-point part labels and the robot state.

In this work, we tackle the problem under the setting of imitation learning, following the setting of the "no external annotation" track in ManiSkill Challenges, where several successful demonstrations are available for each training object instance but no external annotations are allowed. We further get rid of any hand-crafted task-specific reward functions and only assume a terminal reward r_t (1 if success, otherwise 0), aiming to learn a category-level generalizable manipulation policy via imitation learning from demonstrations in a task-agnostic manner. The goal is to learn a policy that can handle both training and unseen novel instances.

4 Method

In this section, we introduce our method in detail. In section 4.1, we briefly introduce our baseline imitation learning method that couples GAIL with SAC; in section 4.2, we further propose to progressively grow the structure of the discriminator of GAIL to mitigate the diminishing reward problem; in section 4.3, we introduce the basic version of our proposed generative adversarial self-imitation learning from demonstrations that combines GAIL and self-imitation learning from demonstrations (Pshikhachev et al., 2021) to better leverage both the knowledge of successful experience and the expert demonstration; in section 4.4, we introduce our last improvement, which maintains an instance-balancing expert buffer for self-imitation from demonstration; the implementation details are elaborated in section A.1.

4.1 GAIL with SAC

Our baseline implements GAIL (Ho & Ermon, 2016) to utilize demonstrations provided by ManiSkill Challenge. These expert demonstrations $\{\tau_E\}$ are generated by training per-instance manipulation policy using a popular off-policy model-free reinforcement learning method, Soft Actor-Critic

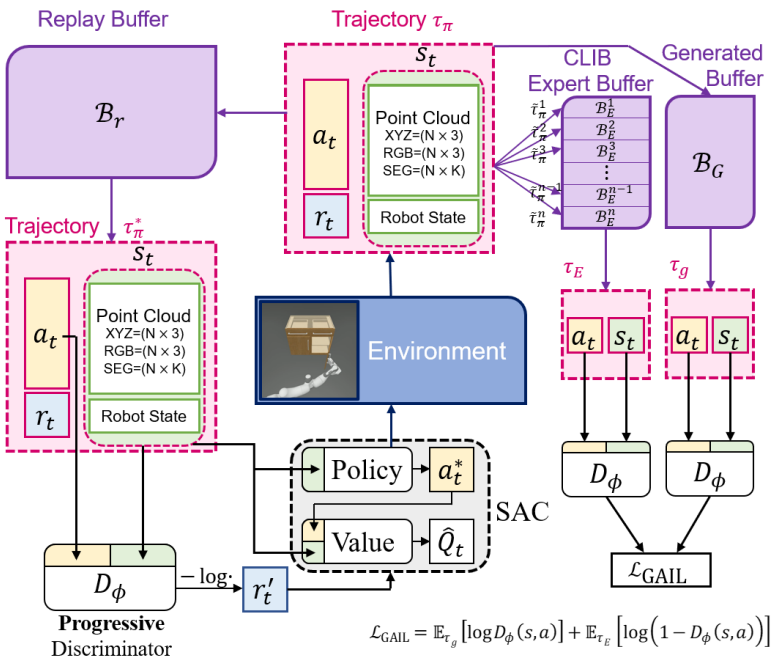

Figure 1: **Pipeline Overview**. On top of Generative Adversarial Imitation Learning, we introduce Category-Level Instance-Balancing (CLIB) Expert Buffer, which both includes expert demonstrations and successful trajectories and maintain a balance between different instances of objects. Besides, we modified the discriminator's structure to make it progressive as training goes.

(SAC) (Haarnoja et al., 2018), with task-specific handcrafted dense rewards. Without using these task-specific rewards, we thus couple GAIL with SAC to provide dense rewards. In this baseline, GAIL utilizes an adversarial trajectory generation scheme that comprises a generator, which is simply the policy network of SAC, and a discriminator D_ϕ, which trains to distinguish between expert trajectories τ_E and generated trajectories τ_g and whose outputs can provide dense rewards r_t' to SAC. This off-policy imitation learning framework contains three buffers: replay buffer $\mathcal{B}_r$, generated buffer $\mathcal{B}_g$ (generated trajectories), and expert buffer $\mathcal{B}_E$. The details are shown in Fig. 1.

4.2 Progressively Growing the Discriminator of GAIL

In GAIL, the reward $r_t' = -\log D_\phi(S_t, a_t)$ is crucial to the success of imitation learning. Due to the need to handle many training instances with large variations, the policy network struggles to ramp up at the very beginning and can easily be over-powered by the discriminator. In other words, the discriminator is usually too strong and can easily tell the generated trajectories from the expert demonstration trajectories soon after the training starts. This will cause the reward from GAIL to quickly vanish to zero, leading to failures in imitation learning.

Inspired by Progressive Growing GAN (PG-GAN) (Karras et al., 2018) and Curriculum GAN (Sharma et al., 2018), we propose to progressively grow the architecture of discriminator during the training, to alleviate the imbalance. More specifically, we let the discriminator evolve from a simple initial architecture to a more complex one. Here we adopt the baseline PointNet architecture proposed by Mu et al. (2021b) as our initial architecture while using the PointNet + Transformer architecture in Mu et al. (2021b) as our final architecture. As shown in Fig. 2, both the initial and final architectures share the same backbone, which comprises K+2 independent PointNet f_i $(i = 1, 2, ..., K+2)$, where K is the number of part masks defined by the task, $f_i(i \leq K)$ extract the feature of part point cloud P_i, f_{K+1} extracts the feature of the rest points, and f_{K+2} extracts the feature of all points P_0 together. Taking these K+2 features along with a trainable embedding vector (serving as a bias for the task) and a processed robot state (we call this K+4 sub-features) as inputs, our initial architecture simply performs a max-pooling along K+4 to generate the final feature, while our final architectures further leverage a Transformer layer to process the features and then perform an attention pooling. When progressive growing is triggered, a parameter α is introduced to linearly interpolate between the output of max pooling and attention pooling(Fig. 2). The network also includes stabilizing phases before and after the growth of α. At the full growth of the progressive network, the structure is the same as PointNet + Transformer structure. In the end, the global feature goes through an MLP and Sigmoid layer to form the output of the discriminator.

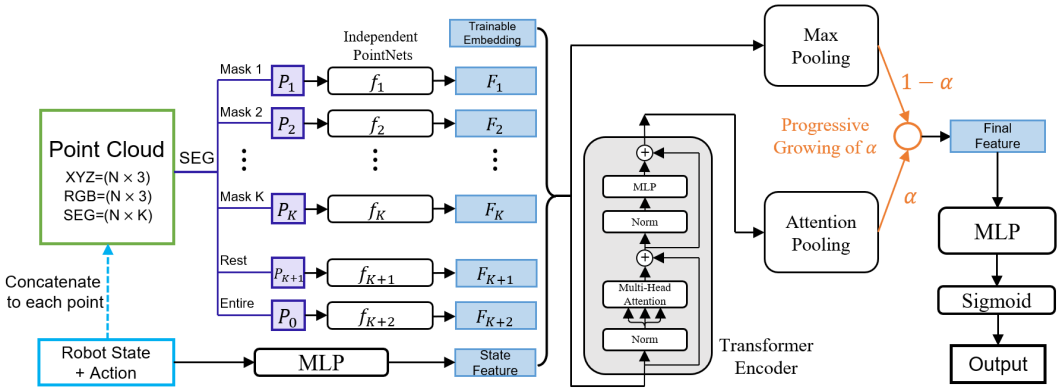

Figure 2: **The progressive structure of discriminator network.** The input contains the point cloud, robot state, and action. The robot state and action are concatenated to a vector in the figure. The output of the network is a scalar ranging from 0 to 1. During training, the latter part of the network progressively grows from max pooling to attention pooling.

The underlying philosophy of our method is indeed curriculum training and echoes Curriculum GAN, which gradually increases the sophistication of the discriminator to make the generator learn smoothly. In our case, the discriminative power of the discriminator is gradually increased and thus the discriminator is more generous with its reward at the beginning of the training than later, allowing the policy network to slowly warm up. Note that we don't evolve the architecture of generator (policy network) based on the following two reasons: 1) the output of our policy network is always in the same action space without any change, which is different from PG-GAN that increases the generation resolution from coarse to fine; 2) since the issue of GAIL comes from the overpowered discriminator, we want the policy network to be as strong as possible from the very beginning.

4.3 Generative Adversarial Self-Imitation Learning from Demonstrations

With the progressive growth of the discriminator, the reward from GAIL will not diminish but we observe that expected reward from GAIL still tends to decay as the training goes (see Fig.). This means the discriminator can be more and more successful to tell apart our generated trajectories from expert demonstration trajectories. One issue we identified that contributes to this problem is the clustering phenomenon in the trajectory space of expert demonstrations. The demonstration provided by ManiSkill challenge is obtained from per-instance RL training, therefore the underlying manipulation strategies can be quite different for different object instances (see Fig. 4(a)). On the other hand, our policy network tends to learn a more universal strategy that can handle different object instances in a similar way (see Fig. 4(b)), which comes by nature from the continuity of neural networks with the change in object instances. In general, expert demonstrations of different object instances may come from different sources and thus exhibit a highly non-uniform distribution in the trajectory space, which is not friendly to imitate using a single policy network.

Inspired by Generative Adversarial Self-Imitation Learning (GASIL) (Guo et al., 2018) and Self-Imitation learning from Demonstration (SILfD) (Pshikhachev et al., 2021), we propose to fill the expert buffer with the successful trajectories generated by our policy network, which is so-called *self-imitation learning*. We thus propose Generative Adversarial Self-Imitation Learning from Demonstrations that combines GASIL, which leverages GAIL with Self-Imitation Learning but does not consider expert demonstrations, and SILfD, which does not use GAIL to generate rewards. More specifically, at the beginning of training, we initialize the expert buffer with expert demonstrations. During training, we select successful trajectories, who obtain the terminal rewards, of current policy through the interaction with the environment and update the expert buffer with those trajectories. As training goes, the self-generated trajectories will gradually occupy the expert buffer and yield a more uniform distribution of trajectory data. With this change, the expected reward increases significantly throughout the training process (see Fig. 3).

4.4 Category-Level Instance-balancing Expert Buffer

Lastly, we propose to evenly divide the expert buffer $\mathcal{B}_E$ into many slots $\{\mathcal{B}_E^j\}$ $(j = 1, ..., n)$ of the same size, where n is the number of training instances. Trajectories of one instance O^j, either from expert demonstrations or self-generated successful ones, can only occupy its corresponding slot $\mathcal{B}_E^j$, yielding our category-level instance-balancing (CLIB) expert buffer. At the beginning of

Table 1: Main Results and Ablation Studies on the four tasks of ManiSkill Challenge Benchmark

Method	GAIL	Progressive Growing of Discriminator	SILFD	CLIB Expert Buffer	Task	OpenCabinet Drawer	OpenCabinet Door	PushChair	MoveBucket	Avg
I	✓				Train	0.43±0.03	0.25±0.03	0.19±0.02	0.16±0.02	0.26±0.03
					Val	0.36±0.03	0.15±0.03	0.11±0.01	0.11±0.02	0.18±0.03
II	✓	✓			Train	0.52±0.03	0.38±0.06	0.21±0.03	0.19±0.03	0.33±0.03
					Val	0.49±0.04	0.27±0.04	0.16±0.04	0.16±0.03	0.27±0.04
III	✓		✓		Train	0.45±0.02	0.25±0.05	0.21±0.03	0.17±0.02	0.27±0.02
					Val	0.41±0.06	0.24±0.03	0.16±0.02	0.14±0.03	0.25±0.02
IV	✓		✓	✓	Train	0.58±0.05	0.37±0.05	0.23±0.05	0.25±0.05	0.36±0.03
					Val	0.62±0.04	0.30±0.04	0.18±0.04	0.16±0.03	0.31±0.03
V	✓	✓	✓	✓	Train	**0.61±0.04**	**0.41±0.05**	**0.27±0.04**	**0.27±0.04**	**0.39±0.03**
					Val	**0.65±0.04**	**0.33±0.05**	**0.24±0.03**	**0.19±0.04**	**0.36±0.04**

the training, we initialize $\mathcal{B}_E^j$ with the expert trajectories of O^j. In each iteration of GAIL, $\mathcal{B}_E^j$ is updated by adding the success trajectories of O^j in a FIFO (First In First Out) manner.

The motivation of this design is to avoid bias in the expert buffer and improve generalization to novel object instances. For training our proposed generative adversarial self-imitation from demonstration, manipulation on some instances may happen to succeed very earlier than on the others and trajectories from these instances will quickly pop up in the expert buffer, leading to a highly biased distribution (see Fig. 5). These instances that succeed in the early stages are usually similar to each other, so GAIL may over-fit in this particular kind of instance, leading to difficulty in learning to manipulate other kinds of object instances. This is basically the Matthew Effect. Also, if some instances are harder than the others and need more time to be harnessed, our CLIB expert buffer will always keep their expert demonstrations until their successful manipulation arrives.

5 Experiment

5.1 Benchmark

We evaluate our methods on ManiSkill Benchmark (Mu et al., 2021b) (ManiSkill). There are four tasks in ManiSkill and these tasks cover different types of object motions. See section A.2 for more details about statistics for the four tasks, the observation and the dense reward of the benchmark.

5.2 Results and Analysis

The results are summarised in Table 1. We evaluated our methods on four tasks via 100 trials with three different random seeds. In Table 1, with [Progressive Growing of Discriminator], [Self-Imitation Learning from Demonstrations] and [CLIB Expert Buffer], our (Method V) I) outperforms [GAIL] (Method I) by **13%** and **18%** averaged across four tasks on training and validation sets. In the following, we will analyze the contribution from each individual technique we introduce.

5.2.1 Effect of the Progressive Growing of Discriminator on top of GAIL

From Table 1 we can see that compared with [GAIL] (Method I), [Progressive Growing of Discriminator](Method II) significantly improves success rate by **7%** and **9%** averaged across four tasks on train and validation sets, respectively. Fig. 3 shows the expert reward curve from the discriminator during training. With a strong discriminator, the expert reward from the discriminator in GAIL will quickly drop to a very low level. This will have a significant negative impact on RL learning, which is largely dependent on the reward's guidance. However, when the discriminator grows from a simpler network to a more sophisticated network, the reward provided by the discriminator is more stable. By progressive growth, the discriminator can be shaped as a more complex structure to produce more comprehensive rewards, which will, in turn, boost the generalization ability of RL. Moreover, by comparing the method of progressively growing both the generator and the discriminator with the method of only growing the discriminator, we find that with the growing generator, the method will decrease by **17%** and **15%** averaged across four tasks on training and validation sets (Table 2), respectively. We consider that different from PG-GAN which increases the generation resolution from coarse to fine, the output of our generator is always in the same action space so we only need a strong generator and gradually increase the sophistication of the discriminator.

Table 2: Ablation Studies on Progressive Growing

Progressive Growing	Task	OpenCabinet Drawer	OpenCabinet Door	PushChair	MoveBucket	Avg
Discriminator	Train	**0.52±0.03**	**0.38±0.06**	**0.21±0.03**	**0.19±0.03**	**0.33±0.03**
	Val	**0.49±0.04**	**0.27±0.04**	**0.16±0.04**	**0.16±0.03**	**0.27±0.04**
Discriminator and Generator	Train	0.25±0.03	0.19±0.03	0.11±0.04	0.10±0.02	0.16±0.02
	Val	0.16±0.03	0.16±0.04	0.10±0.02	0.06±0.01	0.12±0.02

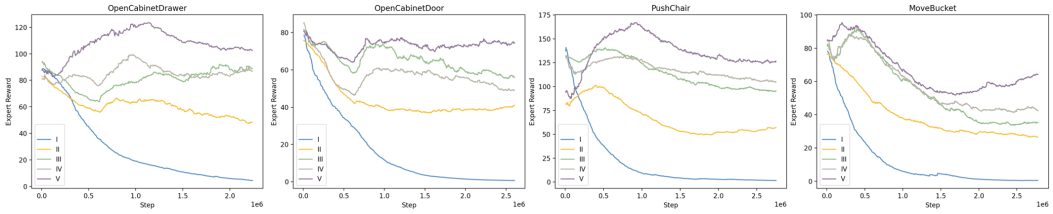

Figure 3: **The expert reward curve during training.** The expert reward comes from the discriminator. When the expert reward is relatively small, it indicates that the discriminator can easily distinguish between the expert data and the data generated by the policy. Notice that our method (curve V) always achieve the highest value.

5.2.2 EFFECT OF THE SILFD ON TOP OF GAIL

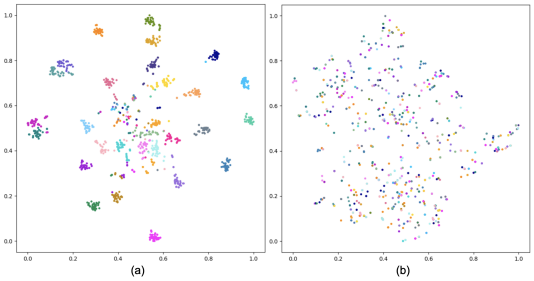

Figure 4: **A t-SNE visual comparison of the feature** extracted by the trained discriminator from the initial data in the expert buffer and the data in the expert buffer at 2×10^6 steps for **MoveBucket** task using our **SILFD** method. (a) initial data in the expert buffer. (b) data in the expert buffer at 2×10^6 steps which are mainly generated by our own policy during training. Each color represents the data of an instance.

From Table 1 we can see that though the performance on training does not show a significant increase, [GAIL] plus [SILFD](Method III) outperforms [GAIL](Method I) by **7%** on the validation sets of four tasks on average. Since the [SILFD] (Method II) updates the expert buffer with the successful trajectories generated by the recent SAC policy, the trajectories in the expert buffer will resemble the generated trajectories. This will let the reward from the discriminator converge to a relatively high value, as shown in Fig. 3. Instead of only imitating from initial expert demonstrations which are circumscribed by the difference between each instance, Self-Imitation Learning provides expert buffer with more various trajectories, thus improving generalization ability. Fig. 4 shows a visual comparison of features using t-SNE between the initial data in the expert buffer and the data in the expert buffer after certain epochs of training. The initial expert data is generated by different single-instance-specialized RL agents on different instances and thus has many separate manifolds in the feature space and can be easily divided into clusters of different instances. On the other hand, the manifold of trajectories generated by our generalized policy is more smooth and continuous and has a well-mixed distribution.

5.2.3 EFFECT OF CLIB EXPERT BUFFER

From Table 1 we can observe that by making the expert buffer balanced during training, our [CLIB Expert Buffer](Method IV) method surpasses [Self-Imitation Learning from Demonstrations](Method III) by **6%** on the validation sets of four tasks on average. Besides, the improvement is significant when the instances vary greatly in size, shapes, etc. like cabinet drawers in **OpenCabinetDrawer** task. Fig. 5 shows the distribution of data in the expert buffer on different instances in the later stage of training. It shows that during training, our [CLIB Expert Buffer](Method IV) method ensures that the data in the expert buffer is evenly distributed on instances while the expert data is much unevenly distributed on instances without using the method. This means the successful trajectories in the expert buffer on some instances may be less and less or even there is no successful trajectory on these instances if the RL hasn't generated successful trajectories on these instances in the early time of training. Thus, the RL may lose the guidance of expert data on these instances. An example of loss of guidance is shown in Fig. 6. Some instances of the drawer with handles that would not easily be grasped by paralleled gripper (ID: 13 & 16) are also hard to be manipulated. Without Category-Level Instance-Balancing (CLIB), these instances' demonstrations will soon be ejected out of expert buffer, so the policy network will have great difficulties in learning how to operate these drawers. However, with the help of CLIB, the expert buffer always reserves demonstrations for these drawers so that the agent also gets some successful knowledge on these instances.

Finally, we combine these methods together. Fig. 3 reveals that when using Self-Imitation Learning from Demonstrations, the reward is higher and stabler than the method that only uses progressive

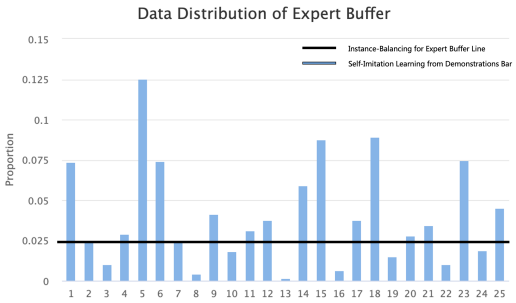

Figure 5: **The data distribution of expert buffer on different instances** at 2×10^6 steps for **OpenCabinetDrawer** task. The black line represents the data distribution of the expert buffer using our **CLIB Expert Buffer** method which is evenly distributed on different instances. The blue bar represents the data distribution of the expert buffer without using this method. The X-axis and Y-axis represent different instances and the proportion of their data in the expert buffer.

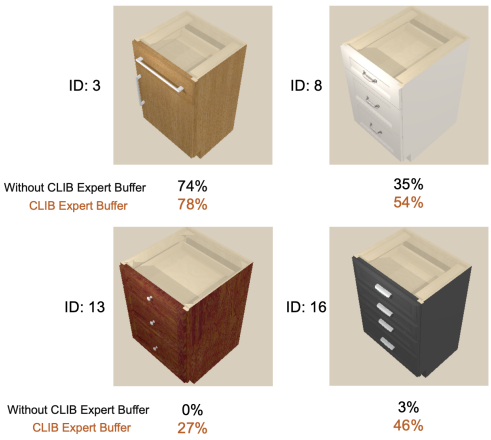

Figure 6: **The success rate on some variant instances before and after using our CLIB expert buffer**. The handles of cabinets have many variations, and some handles, for example, nail shaped, are hard to be opened (ID:13). Without CLIB, the Expert Buffer will soon forget the initial demonstrations for these instances and the agent will not learn to finish these tasks.

growing of discriminator. This occurs since Self-Imitation Learning from Demonstrations will update expert buffer with self policy generated trajectories, which adds difficulties to discriminate between newly generated trajectories and those stored in the expert buffer. Therefore the expert reward remains higher in Self-Imitation Learning from Demonstrations methods. However, the structure of the discriminator does not change in GAIL with Self-Imitation Learning from Demonstration, the discriminator will be misled and gradually loses its discriminative power. The combination of Self-Imitation Learning from Demonstrations, CLIB Expert Buffer, and progressive growing of discriminator can help to solve this problem. Our final method reaches the highest success rate on both training set and validation set, improving **13%** success rate on average training and **18%** success rate on average validation.

5.3 More Results with Handcrafted Dense Reward

Table 3: Additional Experiments with Dense Reward

Method	Task	OpenCabinet Drawer	OpenCabinet Door	PushChair	MoveBucket	Avg
SAC	Train	0.93±0.02	0.69±0.03	0.52±0.02	**0.76±0.04**	0.72±0.04
	Val	0.92±0.02	0.35±0.03	0.35±0.04	0.61±0.04	0.56±0.03
GAIL + Dense Reward	Train	0.97±0.02	0.92±0.02	0.54±0.03	0.50±0.03	0.73±0.04
	Val	0.88±0.01	**0.78±0.02**	0.42±0.04	0.47±0.04	0.63±0.04
V + Dense Reward	Train	**0.99±0.01**	**0.94±0.02**	**0.61±0.04**	0.64±0.05	**0.80±0.04**
	Val	**0.96±0.01**	0.76±0.04	**0.46±0.04**	**0.63±0.04**	**0.70±0.05**

Additionally, we evaluate our methods with additional handcrafted dense reward provided by ManiSkill. The results are summarised in Table 3. We simply combine the expert reward from the discriminator and the environment reward together and use this total reward to update the policy and value networks in RL. We find that our methods with additional dense reward, which ranks the first place on the "no external annotation" track of ManiSkill Challenge 2021, can outperform the GAIL+SAC baseline by **7%** averaged across four tasks on both training and validation sets.

6 Conclusion

In this paper, we, for the first time, tackle the problem of category-level object manipulation via generative adversarial self-imitation learning from demonstrations. We build our method upon GAIL with SAC. We propose several important techniques to improve the baseline, including combining GAIL with self-imitation learning from demonstrations, progressive growing of discriminator, and instance-balancing for expert buffer. Our experiments have shown that our methods can reach a much higher success rate on four manipulation tasks from ManiSkill benchmark than existing baselines and our ablation studies further validates the contribution of each proposed technique.

REFERENCES

Ben Abbatematteo, Stefanie Tellex, and George Konidaris. Learning to generalize kinematic models to novel objects. In *Proceedings of the 3rd Conference on Robot Learning*, 2019.

Miguel Arduengo, Carme Torras, and Luis Sentis. Robust and adaptive door operation with a mobile robot. *Intelligent Service Robotics*, 14(3):409–425, 2021.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pp. 103–129, 1995.

Sachin Chitta, Benjamin Cohen, and Maxim Likhachev. Planning for autonomous door opening with a mobile manipulator. In *2010 IEEE International Conference on Robotics and Automation*, pp. 1799–1806. IEEE, 2010.

Coline Devin, Pieter Abbeel, Trevor Darrell, and Sergey Levine. Deep object-centric representations for generalizable robot learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7111–7118. IEEE, 2018.

Samir Yitzhak Gadre, Kiana Ehsani, and Shuran Song. Act the part: Learning interaction strategies for articulated object part discovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15752–15761, 2021.

Yijie Guo, Junhyuk Oh, Satinder Singh, and Honglak Lee. Generative adversarial self-imitation learning. *arXiv preprint arXiv:1812.00950*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Tatsuya Harada, Antonio Tejero-de Pablos, Stefano Quer, and Francesco Savarese. Service robots: A unified framework for detecting, opening and navigating through doors. In *International Conference on Software Technologies*, pp. 179–204. Springer, 2019.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Ajinkya Jain and Scott Niekum. Learning hybrid object kinematics for efficient hierarchical planning under uncertainty. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5253–5260. IEEE, 2020.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations*, 2018.

S Levine, N Wagener, and P Abbeel. Learning contact-rich manipulation skills with guided policy search (2015). *arXiv preprint arXiv:1501.05611*, 2015.

Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3706–3715, 2020.

Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

Mayank Mittal, David Hoeller, Farbod Farshidian, Marco Hutter, and Animesh Garg. Articulated object interaction in unknown scenes with whole-body mobile manipulation. *arXiv preprint arXiv:2103.10534*, 2021.

Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13001–13011, 2021a.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Cathera Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pp. 3878–3887. PMLR, 2018.

L Peterson, David Austin, and Danica Kragic. High-level control of a mobile manipulator for door opening. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, volume 3, pp. 2333–2338. IEEE, 2000.

Georgiy Pshikhachev, Dmitry Ivanov, Vladimir Egorov, and Aleksei Shpilman. Self-imitation learning from demonstrations. In *Deep RL Workshop NeurIPS 2021*, 2021.

Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 101–103, 1998.

Andreas J Schmid, Nicolas Gorges, Dirk Goger, and Heinz Worn. Opening a door with a humanoid robot using multi-sensory tactile feedback. In *2008 IEEE International Conference on Robotics and Automation*, pp. 285–291. IEEE, 2008.

Rishi Sharma, Shane Barratt, Stefano Ermon, and Vijay Pande. Improved training with curriculum gans. *arXiv preprint arXiv:1807.09295*, 2018.

Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8876–8884, 2019.

Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. *Advances in Neural Information Processing Systems*, 30, 2017.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11097–11107, 2020.

Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 30–37. IEEE, 2016.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv:2010.14406*, 2020a.

Vicky Zeng, Timothy E Lee, Jacky Liang, and Oliver Kroemer. Visual identification of articulated object parts. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2443–2450. IEEE, 2020b.

A APPENDIX

A.1 IMPLEMENTATION DETAILS

A.1.1 TRAINING DETAILS

We call one step a one-time-step interaction with the environment of a single agent, and for each instance task, the max step for interaction is 200. 8 independent interactions run parallel to collect trajectories in a single iteration. For each iteration, SAC networks sample batch size of 1024 from replay buffer and update parameters for 4 times, and discriminator network also sample batch size of 1024 from both Generated Buffer and expert buffer and update parameters for 5 times. The total training step is set to 3×10^6 for all methods. Replay buffer size is set to 6×10^5. For Generative Adversarial Self-imitation Learning from Demonstration, the expert buffer size is set to 6×10^5, and for Category-Level Instance-Balancing expert buffer, each instance expert buffer size is set to 6×10^3. The learning rate for value and discriminator network is 5×10^{-4}, and the learning rate for policy network is 3×10^{-4}.

A.1.2 NETWORK ARCHITECTURE

We use the PointNet model to process point cloud state input for the original discriminator network and PointNet + Transformer model (Mu et al., 2021b) for the policy and value network in the SAC part. The input of the point cloud state consists of the point cloud of the environment and robot state. For the value and discriminator network, the action vector is concatenated to the robot state to be processed together. Then robot state with action is concatenated to each point in the point cloud. This concatenated point cloud is then processed as described in section 4.2 to form K+4 sub-features. For policy and value networks that are not progressive, the K+4 sub-features are sent into PointNet + Transformer and output as a feature vector. The policy network uses MLP to turn the feature into parameters of Gaussian distribution and sample an action vector. The value network uses MLP to turn the feature into an estimated Q value.

In Fig. 2, the progressive growing can be divided into 3 stages. In the first stage, the α remains at 0 for several iterations of training until the network is stabilized. In the second stage, α increases from 0 to 1 linearly as training goes, which will enable the transformer networks to be taken into calculation. In the final stage, α keeps at 1, and the fully grown structure is the same as PointNet + Transformer.

A.2 MORE DETAILS ABOUT BENCHMARK

A.2.1 STATISTICS FOR FOUR TASKS

Table 4: Statistics for Four Tasks

Task	Objects		Dual-arm Collaboration	Action Space Dimension
	Train	Test		
OpenCabinetDrawer	42	10	No	13
OpenCabinetDoor	25	10	No	13
PushChair	26	10	Yes	22
MoveBucket	29	10	Yes	22

There are four tasks in ManiSkill benchmark.

- **OpenCabinetDrawer**: In this task, a single-arm robot is required to open a designed drawer on a cabinet.
- **OpenCabinetDoor**: In this task, a single-arm robot is required to open a designed door on a cabinet.
- **PushChair**: In this task, a dual-arm robot is required to push a swivel chair to a target location on the ground and prevent it from falling over.
- **MoveBucket**: In this task, a dual-arm robot is required to move a bucket with a ball inside and lift it to a platform.

Statistics for the four tasks are summarized in Table 4.

A.2.2 OBSERVATION

The observation of the task is composed of two components: (i) robot state (a vector that describes the agent's state, including pose, velocity, the angular velocity of the moving platform of the robot, joint angles, and joint velocities of all robot joints, positions, and velocities of the robot fingers) ; (ii) point cloud of the scene(6$\pm$k dimensions: 3 XYZ positions for each point, 3 RGB values for each point, k task-relevant segmentation masks).

A.2.3 DENSE REWARD

ManiSkill provides carefully designed well-shaped dense reward functions for each task. Since the dense reward function is expensive to design and is task-specific in the robotic tasks and needs to be manually adjusted for the particular task, we don't assume we have it in the main experiments. However, we provide an evaluation of our methods with an additional handcrafted dense reward section 5.3.

A.2.4 LIMITATIONS

The limitation of applying our methods without any dense reward is revealed in long-horizon tasks. For examples, in tasks of MoveBucket, these tasks require robot to achieve certain states at each stage (e.g. firmly grasp the bucket handle before moving it). Without the guidance from an accurate stage-wise reward function, the overall success rate is not yet perfect. However, our contributions are orthogonal to reward engineering. It is very promising to combine our method with well-shaped dense reward, yielding excellent performance on the tasks.