Learning from Observation-Only Demonstration for Task-Oriented Language Grounding via Self-Examination

Tsu-Jui Fu[†], Yuta Tsuboi[‡], Sosuke Kobayashi[‡], Yuta Kikuchi[‡]

[†]UC Santa Barbara [‡]Preferred Networks, Inc.

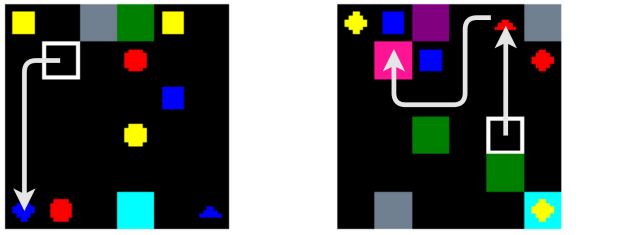
Abstract

Imitation learning is effective to learn control policy from expert demonstrations. Instead of traditional image-guided methods, instruction-oriented learning makes it more flexible and useful in real-world applications. However, most existing imitation methods rely on the assumptions that the action sequences are given in the demonstrations and that the agent can interact with the demonstrations' state to try-and-error, which greatly reduces the practicality of the methods. In this paper, we focus on imitation learning with fixed, observation-only demonstrations where ground truth action sequences are not explicitly given under the visual-language setting. We propose a two-phase method which first imitates the given demonstrations and then further improves the action policy via self-examination, i.e., evaluating explorations over the state space through auto-generated examples. We evaluate our method on pick-and-place tasks and the result shows that the self-examination actually benefits language grounding.

Introduction

Making autonomous agents to perceive the environment and be able to do a series of related action (Russell and Norvig 1995) is one of the most important targets of Artificial Intelligence (AI) system. In order to acquire such complex skills, agents should be provided with some reward indications which represent the desired goal of behavior. In general, there are two kinds of reward indication for an agent to learn from, reward function and expert demonstration. Reward function is widely studied in numerous reinforcement learning research (Tsitsiklis 1994; Sutton et al. 1999) which directly gives the reward feedback of the behavior under the environment. And, the agent tries to maximize the feedback through trial and error, then finally learns from the reward function defined by the expected skill. However, for complex environments, it is difficult to specify the reward function by human knowledge or hand, especially for those tasks where the success is only defined by the final observation.

Learning from expert demonstrations, also known as imitation learning (Schaal 1999; Argall et al. 2009), can avoid the issue of learning from reward function by watching the examples of successful behavior. Imitation learning learns



"move to the blue diamond" *"put the red triangle in the pink box"*

Figure 1: Examples of task-oriented language grounding.

the control policy by following how expert do in the demonstrations which can be divided into behavioral cloning (BC) (Bain and Sammut 1995; Pomerleau 1989) and inverse reinforcement learning (IRL). With sequences of observation, BC tries to learn a function which maps the observation to the same action as the demonstrations. On the other hand, IRL builds a reward function based on the demonstrations to explain the behavior and learns by reinforcement learning. Generative adversarial imitation learning (GAIL) (Ho and Ermon 2016) (Ng and Russell 2000; Fu, Luo, and Levine 2017) is the state-of-the-art imitation learning method which adopts GAN architecture and a discriminator to discriminate a state-action pair is whether from the imitator or from the expert demonstrations. Then, the imitator updates to fool the discriminator so that it can finally do as well as the expert. Though being a great success, most existing imitation learning methods require demonstrations including expert actions between observations. This will restrict the imitator to learn from existing numerous demonstrations which provides observation-only (i.e., frames) information with actions unavailable. For instance, when one watches NBA video to learn how to play basketball, she/he does not know the explicit knowledge of the muscle control as well as the clear action where the players do.

Imitation from observation (IFO) can be considered as a more practical and more natural way as humans does. Different from typical imitation learning, IFO uses the demonstrations but without the actions executed by the expert. Another restriction of IFO is that interacting with demonstra-

Method	Ground-truth Action	Interact with Demo	Instruction-Oriented
GAIL	✓	×	×
BCO	×	×	×
AGILE	×	✓	✓
Ours	×	×	✓

Table 1: Our problem setting between others.

tions' state is invalid. For some IRL methods, they tend to interact with demonstrations' state and observe what will happen to build the target reward function. However, it is impracticable under reality usage since the demonstrations are observation-only which means we cannot step any actions on them. For instance of watching NBA video, it is impossible to let the player jump with his different leg and record as new observations. Behavioral cloning from observation (BCO) (Torabi, Warnell, and Stone 2018a), one of the state-of-the-art methods of IFO, plans to train an inverse dynamic model to infer the missing actions and then adopts BC to learn the mapping function which maps the observation into action as the same as the demonstrations. However, due to the lack of exploration in the environment as BC, BCO can suffer from compounding error caused by covariate shift.

From another point of view, most of the previous task-oriented imitation learning tasks are image-guided (Pathak et al. 2018) which makes it inflexible during inference. To let the agent imitate, we need to prepare an demonstration video in advance with the objects and the scene being very similar to the real executed environment. This kind of image-guided imitation learning limits the practicality and the generalizability of task-oriented imitation learning. To overcome this limitation, instruction-conditioned visual tasks have been proposed in which learned models are successfully generalized to different objects or unseen situations. Agrawal et al. (2015) evaluates on visual question answering task and answer the question with unseen questions or visual input. Hatori et al. (2018) learns a robot arm to pick an object into another place which is specified by a spoken language. We consider that instruction-conditioned settings can make task-oriented imitation learning more flexible and practical where we can assign the agent to target tasks by simply giving instructions instead of preparing demonstraion videos. Adversarial Goal-Induced Learning from Examples (AGILE) (Bahdanau et al. 2019) also focuses on the instruction-oriented imitation learning which tries to learn a reward function from the demonstrations as IRL. However, in order to learn that reward function, AGILE has to interact with the demonstrations and collect the negative examples which violates the restriction of IFO.

In this paper, we aim at visual-language grounding tasks which apply classic pick-and-place problem, as illustrated in Fig. 1. We follow the challenge IFO setting which learns from observation-only demonstrations with ground-truth action being unknown and interacting with demonstrations being invalid under the visual-language setting. The input contains a simulator which supplies an environment that the agent can interact with and numbers of expert demonstrations to learn from. The exploration under visual-language setting is also challenge. Since the simulator can supply a new state but no suitable instruction with, it is unable to explore without a target to achieve. The comparison of the problem setting between ours and previous works is noted at Table. 1.

We introduce a two-phase method, as illustrated in Fig. 3. In the 1st-phase, we adopt BCO to make our target action policy to imitate how demonstrations execute. To overcome the lack of exploration under behavioral cloning, in the 2nd-phase, we train an instruction module to select suitable instruction for the state initialized from the simulator and a reward module to give the reward feedback using demonstrations, what we call self-examination. With the instruction module and reward module, we make IFO task to be able to try-and-error under visual-language setting and improve itself via self-examination. To evaluate the experimental result, we implement a 2D grid-world with kinds of pick-and-place tasks. Our contributions of this paper are twofold:

1. We propose a challenge and practical instruction-oriented IFO problem where the ground-truth action is unknown and interacting with demonstrations is invalid.
2. We present a two-phase method to imitate the demonstrations under IFO and further improve the performance via self-examination.

Related Work

Visual-language Grounding

Visual-language grounding is widely studied and make a number of practical applications. Agrawal et al. (2015) proposes a task that does question answering based on a reference image. Anderson et al. (2018) learns to navigate in the environment according to a series of language command. Yu, Zhang, and Xu (2018) answers the given question via exploring in the environment. (Hatori et al. 2018) performs on the pick-and-place task where the target object is specified by spoken language. Chaplot et al. (2018) moves to the object with a specific shape, color, and type from language instruction in a 3D environment. In our work, we focus on a variant of the classic pick-and-place problem in which all task parameters (object, start location, and end location) are given by a textual language, and learning supervision is provided in the form of demonstrations.

Mapping Instruction into Actions

There are numerous works on learning to map from instructions into action sequences. Artzi and Zettlemoyer (2013) adopts CCG semantic parser to execute natural language instructions. Misra et al. (2014) manipulates the robot arm to interact with the environment based on human instructions. Misra et al. (2015) leverages the environment to induce new lexical entries during the testing time for high-level instructions. Tellex et al. (2011) performs navigation and mobile manipulation from instructions given to autonomous systems. Branavan et al. (2009) applies reinforcement learning to achieve the desired goal from pairs of instructions. Most methods assume that the action sequences are given and learn the mapping between instruction and action in a

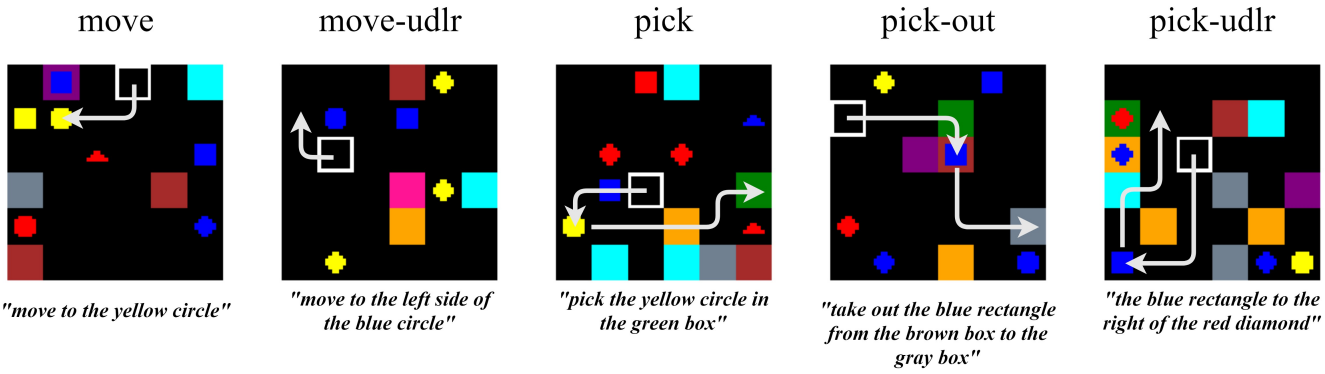


Figure 2: Example of *move*, *move-udlr*, *pick*, *pick-out*, and *pick-udlr* task.

supervised scenario. However, we consider learning from observation-only demonstrations without transition actions known in advance, instead of learning from ground-truth instruction-action pairs directly.

Imitation from Observation

Imitation from observation (IFO) (Liu et al. 2018; Torabi, Warnell, and Stone1 2019; Torabi, Warnell, and Stone 2018b) should be considered for learning from expert more naturally. Compared to typical imitation learning, IFO is more practical and exhibits more similarity with the way many biological agents appear to approach imitation.

Nair et al. (2017) and Pathak et al. (2018) adopt an inverse dynamics model through the self-supervised exploration to infer demonstrations' action. Behavioral cloning from observation (BCO) (Torabi, Warnell, and Stone 2018a) also infers transition actions and mimics demonstrations via behavioral cloning. Despite being able to learn from the observation-only demonstration, all of them are not in a task-oriented scenario. Moreover, being based on behavioral cloning suffers from compounding error due to covariate shift. On the other hand, Adversarial Goal-Induced Learning from Examples (AGILE) (Bahdanau et al. 2019) considers instruction-conditional IFO and learns the agent policy according to a state-reward function trained from expert demonstrations. However, in order to train the reward function, AGILE has to interact with demonstrations' state, step different actions and explore in the environment, which greatly reduces the practicality. LC-RL (Fu et al. 2019) assumes the action is directly observed and also suffers from the difficulty of interacting with the demonstrations where they explore in it to learn the IRL. In additional, exploring is also difficult for the instruction-conditional task because it is unable to do exploration without a suitable instruction for an initial state. In this paper, we present a two-phase method which borrows the advantage from BCO in the 1st-phase and applies an instruction module in the 2nd-phase for selecting suitable instruction, enabling exploration to further improve the performance.

Environment for Proposed Task

We propose a visual-language grounding problem and build an environment with several different tasks for its evaluation.

Overview The environment is target to mimic the pick-and-place task where we have to move the robot arm (the white border) to the specific position or pick an object into the correct place oriented by the instruction. There are 6 actions in our environment:

- U/R/D/L: move the arm up/right/down/left
- P: pick or place the object (pick if the arm is empty, otherwise place the holding object)
- S: stop which means finish the instruction

As Fig. 2, the environment is a 6x6 observable 2D grid-word which is populated with objects of different shapes or colors and boxes (those grids with all color filled in) of different colors. The specification is as following:

- Object Shape: *circle*, *rectangle*, *diamond*, *triangle*
- Object Color: *red*, *yellow*, *blue*
- Box Color: *orange*, *green*, *purple*, *pink*, *brown*, *gray*, *cyan*

All the objects and boxes are randomly placed in the grid-world.

During the evaluation, we will check the final state after the action S or achieving the maximum number (40) of action. The reason why we need a stop action is that for a grounding task, the agent has to know when or what is finish. There can be a probable case that agent achieves the target during the moving process but finally moves out, and it should also be seen as a failure.

Task We propose 5 tasks, *move*, *move-udlr*, *pick*, *pick-out*, and *pick-udlr* for different situations of pick-and-place problems.

- *move*: move the robot arm to the object
- *move-udlr*: move the robot arm to the up/down/left/right side of the object
- *pick*: pick the object into the box
- *pick-out*: take out the object from the box into another box
- *pick-udlr*: pick the object to the up/down/left/right side of another object

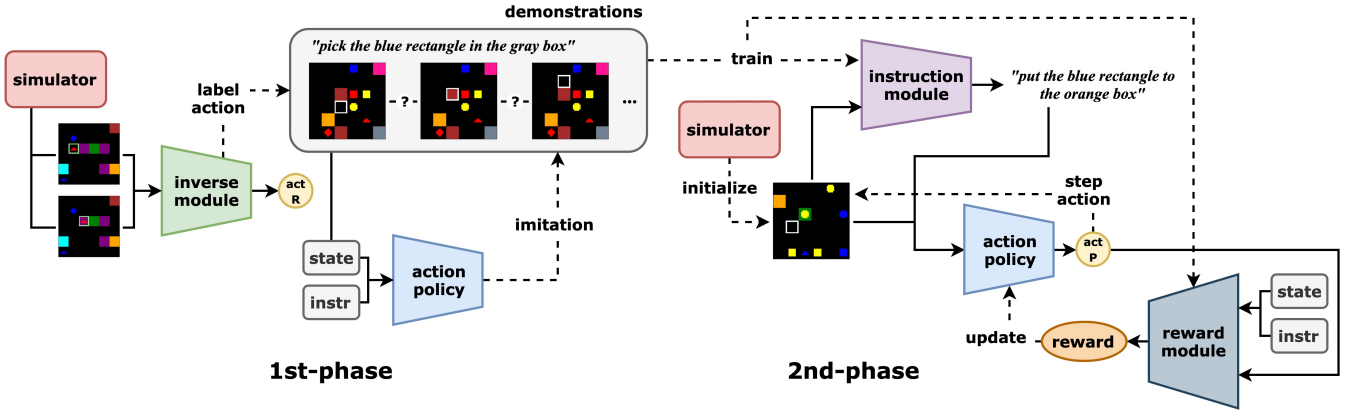


Figure 3: The overview architecture and training flow of both 1st-phase and 2nd-phase.

As BabyAI (Chevalier-Boisvert et al. 2019), the related instructions are generated by templates (1 for *move*, 8 for *move-udlr*, 12 for *pick*, 36 for *pick-out*, 24 for *pick-udlr*) with different object shapes, object colors, and box colors. And the corresponding ground-truth demonstrations are also automatically produced where all paths are the shortest one. To avoid the ambiguity, we make sure that there is only one valid object or box mentioned in the instruction. For example of *pick-out* task in Fig. 2, there are two blue rectangles in the environment but only one in the brown box, therefore, it will not cause any ambiguity.

Problem Definition A demonstration D is composed of consecutive state images $\{s_1, s_2, s_3, ...\}$ and a target instruction t. With the demonstrations $\{D_1, D_2, ...\}$, for each task, we have to train an action policy ϕ to decide an action a to step, given the current state s and instruction t.

Methodology

The overall architecture of our proposed method which contains 2 phases training is illustrated in Fig. 3. In the 1st-phase, motivated by BCO (Torabi, Warnell, and Stone 2018a), we first randomly step the simulator and collect numerous state-action pairs to train an inverse module. Then, we apply the inverse module to infer the unknown action sequences of demonstrations. Finally, we can make the action policy to imitate the demonstrations by predicting the action labeled using the inverse module, given the state image and the target instruction.

In the 2nd-phase, in order to overcome the lack of exploration under behavioral cloning, we train an instruction module and reward module using demonstrations. The instruction module is used to select a suitable instruction for a state initialized from the simulator, providing the action policy a goal to achieve and explore by. The reward module gives out the reward feedback between an action and state-instruction pair, then update the action policy. With the instruction module and reward module, we make IFO task to be able to try-and-error under visual-language setting and improve itself via self-examination.

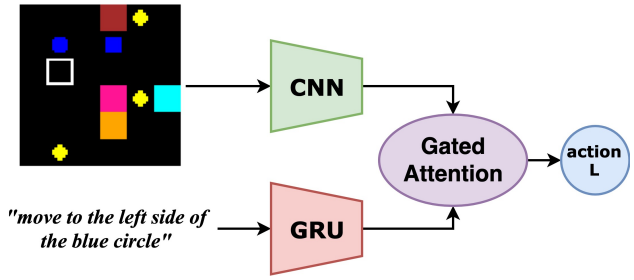


Figure 4: The architecture of our action policy which adopts gated-attention as (Chaplot et al. 2018).

1st-phase

As the left part of Fig. 3, the 1st-phase trains an inverse model to label the actions in demonstrations where the action policy ϕ can behavioral cloning from.

Inverse Module We first collect numerous state-action pairs (s_i, s_{i+1}, a_i) by randomly step the simulator. Then, we can train the inverse module inv supervisely as:

$$\mathrm{inv}(s_i, s_{i+1}) = a',$$

where a' should be a_i. With the inverse module, the missing actions in the demonstrations can be inferred by inv prediction.

Imitation from Demonstration With actions labeled by inv, we apply behavioral cloning to train an action policy ϕ which imitates how demonstrations step action:

$$\phi(s, t) = a',$$

where s is the state image, t is the target instruction, and a' should be as the same as labeled action.

As Fig. 4, we adopt gated-attention (Chaplot et al. 2018) between the state image and the instruction to enhance the performance of inv as following:

$$\mathrm{GA}(s, t) = \mathrm{CNN}(s) \odot h(\mathrm{GRU}(t)),$$

where CNN extracts the visual feature of the state image, GRU models the feature of the input instruction, and h is

a linear layer with sigmoid activation. Assuming the CNN feature size d_s and shape $H \times W$, h will project the GRU feature into size d_s and then extend it into $H \times W$ so that it can do Hadamard product with CNN feature which attends the instruction on specific attributes of the image. The fusion feature from gated-attention then is used to predict the action by several linear layers.

2nd-phase

Although the 1st-phase is able to imitate the demonstrations, the lack of exploration can suffer from compounding error caused by covariate shift. However, exploration in the environment under visual-language setting is not trivial. Since the simulator can only initialize the state, it is hard to explore with the target instruction unknown.

The 2nd-phase trains an instruction module ins to select a suitable instruction for an initial state and a reward module rwd to give out the feedback for updating the action policy.

Instruction Module For a state initialized from the simulator, the instruction module ins selects a suitable instruction from the instruction pool:

$$\mathrm{ins}(s, \{t_1, t_2, ...\}) = t,$$

where $\{t_1, t_2, ...\}$ is the instruction pool which consists of instructions in the demonstrations. To make the target instruction robust, we choose to select an instruction instead of generating one.

The instruction module ins is a binary classifier which judges the suitability of a state-instruction pair:

$$\mathrm{ins}(s, t) = [0, 1].$$

We consider the state-instruction pairs in the demonstrations as the positive pairs and the different instructions chosen randomly as the negative pairs. We also remove out the probable false-negative pairs as (Bahdanau et al. 2019) and train ins using binary cross-entropy loss. And we select an instruction for a state as a suitable one if the output of ins more than 0.8. If there is no one suitable, we will let the simulator initialize a new state and select a suitable instruction again. The architecture of ins also adopts the gated-attention.

Reward Module The reward module rwd is to judge an action is correct for a state-instruction pair. As instruction module ins, rwd is also a binary classifier:

$$\mathrm{rwd}(s, t, a) = [0, 1].$$

We consider the actions labeled by the inverse module inv in the demonstrations as the positive cases and randomly different actions as negative cases during pre-training. Then, rwd is trained with the following self-examination where the actions from action policy ϕ are viewed as the negative cases. rwd also applies gated-attention and is trained using binary cross-entropy loss.

Improvement via Self-Examination With the instruction module inv and the reward module rwd, we can further improve the action policy ϕ in a self-examination scenario.

Firstly, we let the simulator initialize a new state s and select an instruction t by the instruction module ins. Then, we apply action policy ϕ to predict an action a to step and rollout. After rollout, we have a complete execution trajectory $T = \{(s_1, t, a_1), (s_2, t, a_2), (s_3, t, a_3)...\}$. We adopt the reward module rwd to give reward feedback for each step. Finally, we can update ϕ by Policy Gradient (Sutton et al. 1999) as typical reinforcement learning.

By repeating the above process, we can try-and-error with different states and different instructions under the visual-language setting, further improving the target action policy ϕ. The procedure of the self-examination is described in Algorithm 1.

Algorithm 1 self-examination of 2nd-phase

ϕ: action policy
ins: instruction module
rwd: reward module

while self-examination **do**
  $s \leftarrow$ the simulator initializes a new state
  $t \leftarrow$ select a suitable instruction by ins
  $\{(s_1, t, a_1), (s_2, t, a_2), ...\} \leftarrow$ adopt ϕ to rollout (s, t)
  $\{(s_1, t, a_1, r_1), (s_2, t, a_2, r_2), ...\} \leftarrow \mathrm{rwd}$ gives reward
  update ϕ with reward pairs via Policy Gradient
end while

Experimental Results

In this section, we present the experimental results of our proposed method for *move*, *move-udlr*, *pick*, *pick-out*, and *pick-udlr* task. We first describe the experimental settings and show the quantitative results of both 1st-phase and 2nd-phase. Then, we investigate the performance of our instruction module for 2nd-phase and the method generalizability under zero-shot setting. Finally, we demonstrate the execution and the improved effect of 2nd-phase via case study.

Experimental Settings

We evaluate our proposed method on *move*, *move-udlr*, *pick*, *pick-out*, and *pick-udlr* tasks, which are described in above section. There are 80K demonstrations of each task for 1st-phase imitation. For 2nd-phase, we adopt the instruction module to select 10K suitable pairs of state-instruction for the following exploration.

In our implementation, we apply a 4-layer convolutional neural network (CNN) with kernel size 3, feature size 32, stride 2, and padding 1 to extract the visual feature of the state image. We adopt bidirectional Gated Recurrent Units (Chung et al. 2014) (bi-GRU) with hidden size 64 to model the input instruction and the word embedding with size 8 is randomly initialized, then trained with the whole network. During training, we set the dropout rate 0.25, the learning rate of 1st-phase 8e-4, and the learning rate of 2nd-phase 3e-5. We train our method using Adam optimizer (Kingma and Ba 2015) and implement it under PyTorch. For the baseline, we see 1st-phase as the same as BCO (Torabi, Warnell, and Stone 2018a), which is the state-of-the-art method of IFO with ground-truth action unknown and interacting with demonstrations being invalid.

Task	1st-phase	2nd-phase [5K]	2nd-phase [10K]
move	76.6%	**79.4% (+2.8)**	78.4% (+1.8)
move-udlr	69.0%	**73.4% (+4.4)**	72.4% (+3.4)
pick	13.0%	15.0% (+2.0)	**15.4% (+2.4)**
pick-out	32.8%	34.0% (+1.2)	**34.6% (+1.8)**
pick-udlr	22.6%	**26.0% (+3.4)**	24.8% (+2.2)

Table 2: Success rate of both 1st-phase and 2nd-phase for different tasks.

Task	Instruction Suitable Rate
move	91.725%
move-udlr	81.145%
pick	90.190%
pick-out	77.940%
pick-udlr	76.810%

Table 3: Suitable rate of instructions selected by our ***instruction module*** for different tasks.

Quantitative Results

Table. 2 shows the success rate of both 1st-phase and 2nd-phase. For the baseline BCO, which is also the 1st-phase, it achieves 76.6% and 69.0% for simpler task *move* and move-udlr, and 13.0%, 32.8%, and 22.6% for more difficult task *pick*, *pick-out*, and *pick-udlr*, respectively. For 2nd-phase, we select 5K or 10K state-instruction pairs for exploration. The result shows that no matter 5K or 10K, 2nd-phase improves 1.2-4.4 points of success rate which means exploration in the environment actually benefits to all tasks. It also shows that not more state-instruction pairs always conduce to better improvement which is task-dependent and balanced between the imitation (1st-phase) and the exploration (2nd-phase).

Detailed Analysis

Selected Instructions Suitability To make sure the instructions selected by our instruction module are suitable for the initial state during 2nd-phase exploration, we calculate the suitable rate of state-instruction pair. An instruction can be seen as suitable if and only if all instruction-mentioned objects also exist in the state, no ambiguity, and the target can be achieved successfully. Some case studies can be seen in following section.

As shown in Table. 3, the suitable rate for all tasks is larger than 76% which means the instruction module actually selects suitable instructions for initial states from the simulator, making it possible to explore in the environment under instruction-oriented task. Our instruction module gives IFO task an ability to try-and-error under vision-and-language setting, and then the reward module sends out the feedback to further improve the action policy.

Zero-shot Generalization To investigate the generability of visual-language grounding, we evaluate under zero-shot setting where new combinations of attribute-object pairs are unseen in the training instructions. For example, there are *red circle* and *yellow rectangle* in the training but contains *yellow circle* in the testing instructions. We also adopt 80K as demonstrations for 1st-phase and select 5K as exploration instructions for 2nd-phase.

Task	1st-phase	2nd-phase
move	62.8%	**68.4% (+5.6)**
move-udlr	56.8%	**60.6% (+3.8)**
pick	11.2%	**14.6% (+3.4)**
pick-out	27.4%	**29.8% (+2.4)**
pick-udlr	19.8%	**21.0% (+1.2)**

Table 4: Success rate under ***zero-shot*** setting of both 1st-phase and 2nd-phase for different tasks.

Table. 4 shows the result. Even if under zero-shot setting, we can see that there is only little performance drop for 1st-phase. And the 2nd-phase exploration can improve 1.2%-5.6% success rate over 1st-phase as the normal setting.

Case Study

Fig. 5 demonstrates some suitable and unsuitable cases for our instruction module. In general, we can see that all mentioned objects in the instruction also exist in the state. There are also some difficult situations where our instruction module cannot handle perfectly. For instance, the ambiguity (*move*), the target position being occupied or out-of-boundary (*move-udlr*, *pick*, *pick-udlr*), or being unachievable (*pick-out*). However, we argue that the instruction module is still effective. At first, according to Table. 3, most of the selected instructions are suitable for the initial state which gives the action policy a correct goal to achieve. Moreover, though the target positions are unachievable for some cases, the reward feedback during 2nd-phase also benefits to action policy which improves the final success rate.

The executions in the comparison between 1st-phase and 2nd-phase are shown in Fig. 6. Since the action procedures in the demonstrations are all shortest path, the 1st-phase can only execute as shortest. While, due to the opportunity of exploration, the 2nd-phase is able to execute as a different path but also achieve the target successfully (the cases on the first row). In addition, the 1st-phase imitates how demonstrations step and only sees the perfect actions which lead to being stuck and stepping back and forth repeatedly. However, after the try-and-error over the environment during 2nd-phase, the action policy can achieve the goal (the cases on the second row). Apart from the stuck problem, the 2nd-phase also benefits to the problem of the wrong target object or the wrong target position (the cases on the third row).

Conclusion and Future Work

We aim at imitation from observation-only demonstrations with ground-truth action being unknown and interacting with demonstrations being invalid under visual-language setting. We propose a two-phase method which first imitates how the demonstrations step. In the 2nd-phase, we select suitable instructions for initial states and do self-examination to further improve the action policy. The result shows that the 2nd-phase outperforms the baseline by 1.2-4.4 point margins of success rate for several tasks. Although we validate our method on the virtual environment, further study is needed for real-world applications, such as voice-control robot learning via human demonstration.

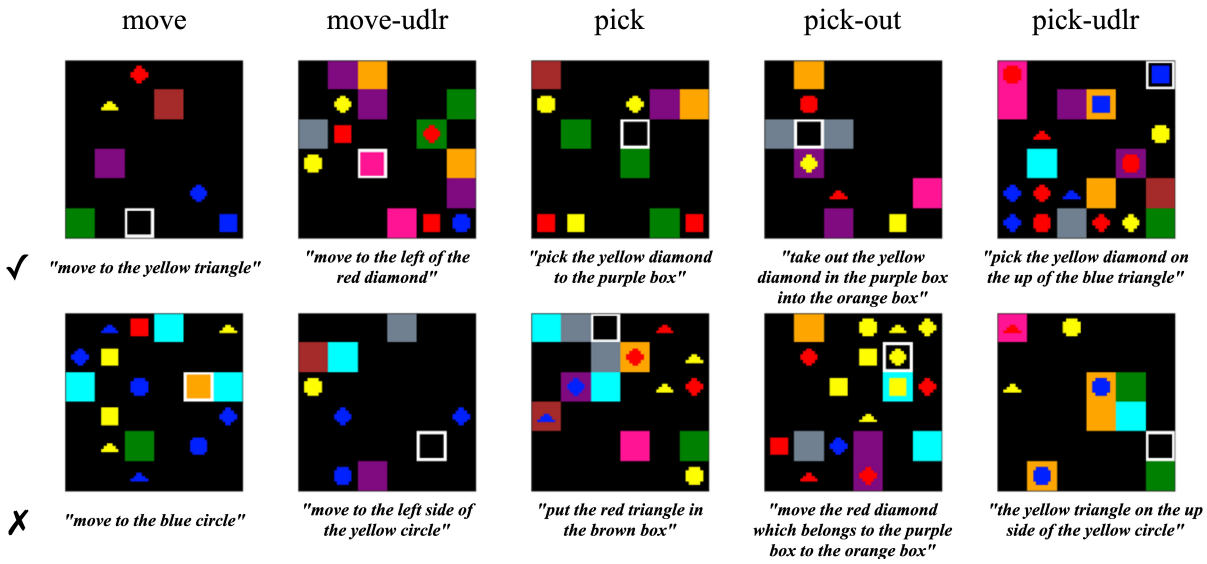


Figure 5: Case study of instructions from the ***instruction module***.

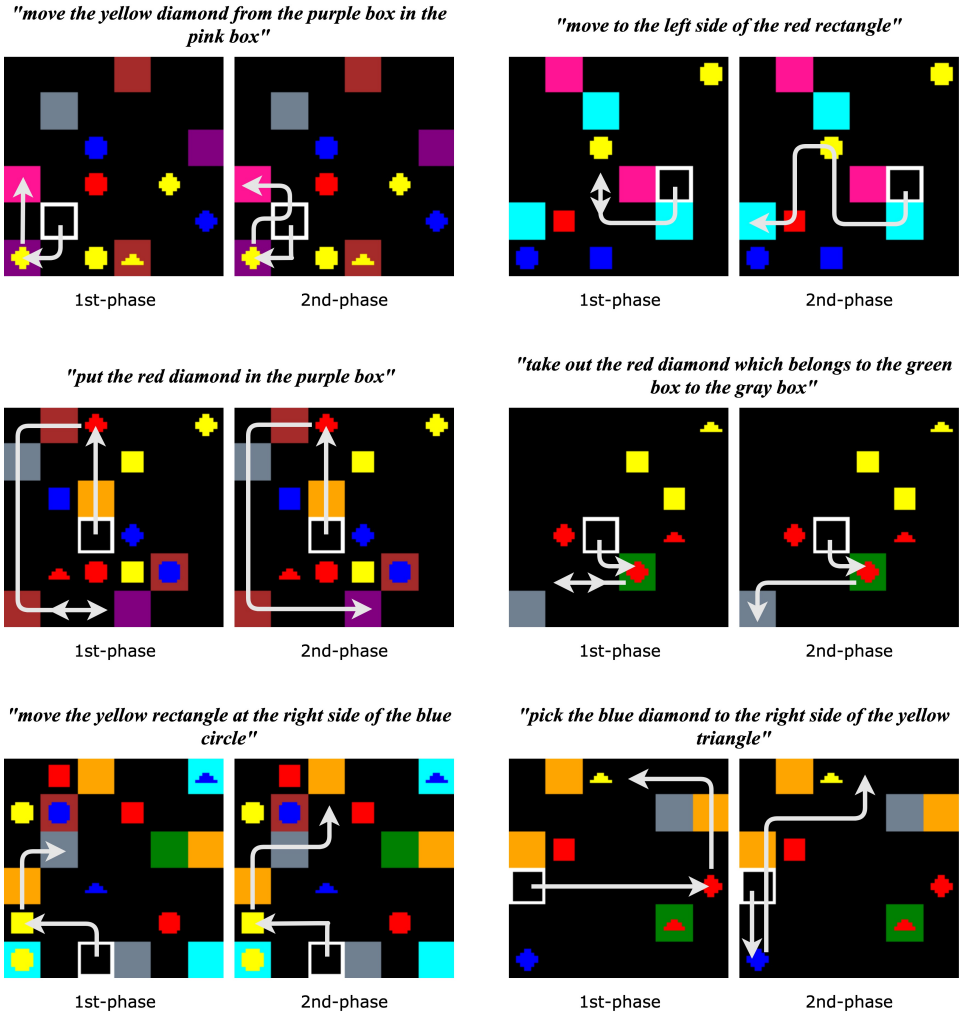


Figure 6: Case study of execution in the comparison between 1st-phase and 2nd-phase.

References

Agrawal, A.; Lu, J.; Antol, S.; Mitchell, M.; Zitnick, C. L.; Batra, D.; and Parikh, D. 2015. VQA: visual question answering. In *ICCV*.

Anderson, P.; Wu, Q.; Teney, D.; Bruce, J.; Johnson, M.; Snderhauf, N.; Reid, I.; Gould, S.; and van den Hengel, A. 2018. Vision-and-language navigation: interpreting visually-grounded navigation instructions in real environments. In *CVPR*.

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. In *Robot. Auton. Syst.*

Artzi, Y., and Zettlemoyer, L. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. In *TACL*.

Bahdanau, D.; Hill, F.; Leike, J.; Hughes, E.; Hosseini, A.; Kohli, P.; and Grefenstette, E. 2019. Learning to understand goal specifications by modelling reward. In *ICLR*.

Bain, M., and Sammut, C. 1995. A framework for behavioural cloning. In *Machine Intelligence*.

Branavan, S.; Chen, H.; Zettlemoyer, L. S.; and Barzilay, R. 2009. Reinforcement learning for mapping instructions to actions. In *ACL*.

Chaplot, D. S.; Sathyendra, K. M.; Pasumarthi, R. K.; Rajagopal, D.; and Salakhutdinov, R. 2018. Gated-attention architectures for task-oriented language grounding. In *AAAI*.

Chevalier-Boisvert, M.; Bahdanau, D.; Lahlou, S.; Willems, L.; Saharia, C.; Nguyen, T. H.; and Bengio, Y. 2019. BabyAI: first steps towards grounded language learning with a human in the loop. In *ICLR*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Workshop*.

Fu, J.; Korattikara, A.; Levine, S.; and Guadarrama, S. 2019. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *ICLR*.

Fu, J.; Luo, K.; and Levine, S. 2017. Learning robust rewards with adverserial inverse reinforcement learning. In *ICLR*.

Hatori, J.; Kikuchi, Y.; Kobayashi, S.; Takahashi, K.; Tsuboi, Y.; Unno, Y.; Ko, W.; and Tan, J. 2018. Interactively picking real-world objects with unconstrained spoken language instructions. In *ICRA*.

Ho, J., and Ermon, S. 2016. Generative adversarial imitation learning. In *NIPS*.

Kingma, D. P., and Ba, J. 2015. Adam: a method for stochastic optimization. In *ICLR*.

Liu, Y.; Gupta, A.; Abbeel, P.; and Levine, S. 2018. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *ICRA*.

Misra, D. K.; Sung, J.; Lee, K.; and Saxena, A. 2014. Tell me dave: context-sensitive grounding of natural language to manipulation instructions. In *RSS*.

Misra, D. K.; Tao, K.; Liang, P.; and Saxena, A. 2015. Environment-driven lexicon induction for high-level instructions. In *ACL*.

Nair, A.; Chen, D.; Agrawal, P.; Isola, P.; Abbeel, P.; Malik, J.; and Levine, S. 2017. Combining self-supervised learning and imitation for vision-based rope manipulation. In *ICRA*.

Ng, A. Y., and Russell, S. J. 2000. Algorithms for inverse reinforcement learning. In *ICML*.

Pathak, D.; Mahmoudieh, P.; Luo, G.; Agrawal, P.; Chen, D.; Shentu, Y.; Shelhamer, E.; Malik, J.; Efros, A. A.; and Darrell, T. 2018. Zero-shot visual imitation. In *ICLR*.

Pomerleau, D. A. 1989. ALVINN: an autonomous land vehicle in a neural network. In *NIPS*.

Russell, S. J., and Norvig, P. 1995. Artificial intelligence: A modern approach. In *Prentice-Hall*.

Schaal, S. 1999. Is imitation learning the route to humanoid robots? In *Trends in Cognitive Sciences*.

Sutton, R. S.; McAllester, D.; Singh, N. S.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M. R.; Banerjee, A. G.; Teller, S.; and Roy, N. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*.

Torabi, F.; Warnell, G.; and Stone1, P. 2019. Recent advances in imitation learning from observation. In *IJCAI*.

Torabi, F.; Warnell, G.; and Stone, P. 2018a. Behavioral cloning from observation. In *IJCAI*.

Torabi, F.; Warnell, G.; and Stone, P. 2018b. Generative adversarial imitation from observation. In *arXiv preprint arXiv:1807.06158*.

Tsitsiklis, J. N. 1994. Asynchronous stochastic approximation and q-learning. In *Machine Learning*.

Yu, H.; Zhang, H.; and Xu, W. 2018. Interactive grounded language acquisition and generalization in a 2D world. In *ICLR*.