A SELF-SUPERVISED METHOD FOR MAPPING INSTRUCTIONS TO ROBOT POLICIES

Hsin-Wei Yu* Po-Yu Wu* Chih-An Tsao* You-An Shen* Shih-Hsuan Lin*
Zhang-Wei Hong Yi-Hsiang Chang Chun-Yi Lee

Elsa Lab, Department of Computer Science, National Tsing-Hua University, Hsinchu, Taiwan
{hsinweiyo,bwoyu85928,hl6540,sliz97028,jerrylin1121}@gmail.com
{williamd4112,ebola777,cylee}@gapp.nthu.edu.tw

ABSTRACT

In this paper, we propose a modular approach for separating the instruction-to-action mapping procedure into two separate stages. The two stages are bridged via an intermediate representation called a *goal*. The first stage maps an input instruction to its corresponding *goal*, while the second stage maps the *goal* to an appropriate policy selected from a set of robot policies. The policy is selected with an aim to guide the robot to reach the *goal* as close as possible. We implement the above two stages as a framework consisting of two distinct modules: an instruction-goal mapping module and a goal-policy mapping module. We evaluate the effectiveness of our method on a number of OpenAI Gym robotic arm manipulation tasks against several baseline methods. Our results show that the proposed framework is able to learn an effective instruction-to-action mapping procedure in an environment with a given instruction set more efficiently than the baseline methods. In addition to the impressive data-efficiency, the results also demonstrate that our modular framework can be adapted to new instruction sets and new robot action spaces much faster than the baselines.

1 INTRODUCTION

Understanding human instructions and interpreting them into actions have long been a crucial need and research focus for autonomous robots (Winograd, 1972; Thomason et al., 2017). Traditionally, this is treated by researchers as a semantic-parsing and instruction-to-action mapping problem. A number of earlier approaches (Chen & Mooney, 2011; Matuszek et al., 2010; 2013; Mei et al., 2016) propose to learn semantic parsers that map natural languages or human instructions to sequences of actions executable by robots in supervised fashions. Although these approaches have been well explored and mostly validated in their specific problem domains, they typically require a significant amount of human supervision and/or linguistic knowledge. In case that the annotated action sequences of a robot or the relevant linguistic background is not directly available, nevertheless, developing such an instruction-to-action mapping procedure becomes no longer straightforward.

Researchers in recent years have attempted to relax the above constraints by adopting two alternative approaches: reinforcement learning (RL) and imitation learning (IL). Both approaches seek to learn a behavior policy π for an agent that executes the given instructions appropriately. Several RL-based techniques have been explored (Branavan et al., 2009; Misra et al., 2017; Hermann et al., 2017; Chaplot et al., 2018). On the other hand, several IL-based techniques have also been leveraged in autonomous robots for executing a diverse range of instructions in a number of tasks (Argall et al., 2009). Although these RL- and IL-based approaches enable robots to learn effective π's to deal with the instruction-to-action mapping problem with little human supervision, they usually train their π's in an end-to-end manner. This prevents π from easy adaption to new sets of instructions or robots with different action spaces, unless dedicated training data are provided additionally.

To deal with the issues mentioned above, we propose a modular approach which separates the instruction-to-action mapping procedure into two separate stages. The two stages are bridged via an intermediate representation called a *goal*. The first stage maps an input instruction $c \in C$ (where C is an instruction space) to a *goal* $g \in \mathcal{G}$ (where $\mathcal{G}$ is the *goal* space), while the second stage maps

*Indicates equal contribution.

g to an appropriate policy π executable by the robot. The policy π is trained with an aim to guide the robot to reach g as close as possible. Different from the previous end-to-end training methods, these two stages are trained separately as long as they both agree on the same *goal* representation. The modular approach allows either of the two stages to be replaced by another implementation. The modular nature of our approach enables a robot to adapt to a new instruction set quickly, and allows different robots to be maneuvered by the same instruction-to-goal mapping function.

In this paper, we implement the above two stages as a framework consisting of two distinct modules: an instruction-goal mapping module and a goal-policy mapping module. The first module embraces the concept of metric learning, which aims to train a metric function called the *distance function* as a non-linear regressor to evaluate whether a given instruction c is close to a *goal* g or not. The second module is implemented as a policy π conditioned on g. Given an input instruction c in the evaluation phase, the instruction-goal mapping module first translates c to a robot-interpretable *goal* g. The translation is performed by using the distance function output as a metric to search for the most similar g in the goal space $\mathcal{G}$. The implementation of the goal-searching procedure is dependent on the nature of the *goal*. For instance, a straightforward enumeration algorithm is likely an appropriate option for a discrete *goal*, while a more complicated sampling-based search scheme is required for a *goal* in the continuous space. In this work, we consider *goals* in the continuous space. Once a *goal* $\hat{g} \in \mathcal{G}$ is derived by the instruction-goal mapping module, it is fed into the goal-policy mapping module as the condition for the conditional policy π to maneuver the robot.

We perform extensive experiments on multiple OpenAI Gym (Brockman et al., 2016) robotic arm manipulation task environments simulated by the MuJoCo physics engine (Todorov et al., 2012), including *FetchReach*, *FetchPush*, *FetchPickAndPlace*, and *FetchSlide*. We compare our framework against an RL-based (*et al.*, 2014) and an IL-based baselines (Bain & Sammut, 1999). Our experimental results show that the proposed framework is able to learn an effective instruction-to-action mapping procedure in an environment with a given instruction set more efficiently than the baseline methods. In addition to the impressive data-efficiency, the results also show that our framework can be adapted to a new instruction set and a new robot action space much faster than the baseline methods. The main contributions of the paper are summarized as the following:

- A modular framework for mapping human instructions to robot policies via a *goal* representation.
- An unsupervised training method requiring neither action labels nor prior instruction-goal pairs.
- A distance function and a policy mapping method for relating an instruction and a *goal*.
- A comprehensive comparison of the learning efficiency between our framework and the baselines.
- A detailed evaluation for the adaptability of our framework against the baseline methods.

The rest of this paper is organized as follows. Section 2 presents the proposed framework. Section 3 demonstrate the experimental results and discusses their implications. Section 4 concludes the paper.

2 Proposed Methodology

In this section, we present the proposed modular framework and the implementation details. We first provide an overview of the framework, followed by an in-depth explanation of the modules in it.

2.1 Overview of the Proposed Framework

Fig. 1 illustrates the proposed framework, which targets at mapping an instruction code $c \in C$ to a behavior policy $\pi \in \Pi$ of a robot, where C and Π represent the instruction space and the set of policies owned by the robot, respectively. The framework consists of two modules in the evaluation phase (the blue parts): an instruction-goal mapping module and a goal-policy mapping module. The former first maps a given instruction c to a goal $\hat{g} \in \mathcal{G}$ (where $\mathcal{G}$ is the goal space), while the latter maps $\hat{g}$ to a conditional policy $\pi(a|o, \hat{g})$ that leads the robot to a *goal* as close to $\hat{g}$ as possible. The main objective of the instruction-goal mapping module is to learn a mapping function between the two spaces C and $\mathcal{G}$. On the other hand, the goal-policy mapping module aims to learn a proper conditional policy $\pi(a|o, g)$ such that the distance between $\hat{g}$ and the actually reached *goal* by the robot is minimized. The instruction-goal and goal-policy mapping modules are trained separately.

In the proposed framework, two key components are trained in the training phase: the distance function $Dist_\phi(c, g)$ (the red part) and the conditional policy $\pi(a|o, g)$ (the green part). The former is utilized in the instruction-goal mapping module, while the latter is employed by the instruction-goal mapping module, as depicted in Fig. 1. The distance function $Dist_\phi(c, g)$ is implemented as a deep neural network (DNN) parameterized by a set of trainable parameters ϕ, and is trained to evaluate

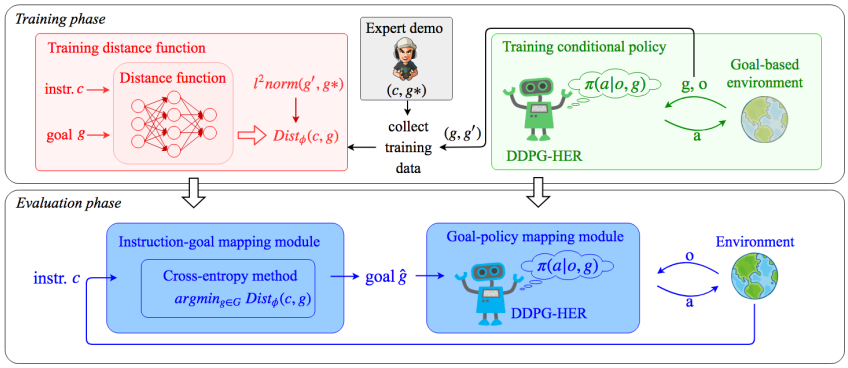

Figure 1: Overview of the proposed framework. During the training phase, we first train DDPG-HER (Andrychowicz et al., 2017) in a goal-based environment. In such an environment, we collect the *desired goal* g and *achieved goal* g' explored by the well-trained conditional policy $\pi(a|o, g)$, as well as the instruction-goal pair $(c, g*)$ demonstrated by an expert for training the distance function $Dist_\phi(c, g)$. In the evaluation phase, we feed the input instruction c to the instruction-goal mapping module, which uses $Dist_\phi$ to search for the closest *goal* $\hat{g}$ to the instruction c by CEM (Rubinstein & Kroese, 2013). The goal $\hat{g}$ is passed to the goal-policy mapping module as the condition for the policy $\pi(a|o, g)$ to interact with the environment.

the affinity between a given instruction c and a *goal* g. The conditional policy $\pi(a|o, g)$ is also implemented as a DNN, and is trained by a method called DDPG-HER (Andrychowicz et al., 2017). During the training phase, the framework first trains the conditional policy $\pi(a|o, g)$ such that for any *desired goal* g, the *achieved goal* g' actually reached by $\pi(a|o, g)$ is close to g as much as possible. The framework then uses the pair (g, g') as well as the instruction-goal pair $(c, g*)$ demonstrated by an expert for training $Dist_\phi(c, g)$. Please note that $g*$ is the ground truth *goal* of instruction c. In the evaluation phase, the instruction-goal mapping module first maps an input instruction c to its closest goal $\hat{g}$ by iteratively searching for $\hat{g}$ in $\mathcal{G}$ using the cross-entropy method (CEM) (Rubinstein & Kroese, 2013), which employs $Dist_\phi(c, g)$ as the metric function for optimization. The goal $\hat{g}$ is then passed to the goal-policy mapping module for $\pi(a|o, g)$ to maneuver the robot to its destination.

2.2 Distance Function

The distance function $Dist_\phi(c, g)$ serves as a metric function for measuring the affinity between any given pairs of c and g. Instead of directly comparing the representations of c and g, $Dist_\phi(c, g)$ is trained to predict the distance between $g*$ and g'. A number of 4-tuples $(c, g, g', g*)$ are first sampled from the expert, the environment, and the conditional policy $\pi(a|o, \hat{g})$. The distance d of each sampled pair $(g', g*)$ is then measured in terms of the L^2 distance, and is formulated as follows:

$$d(g', g*) = \|g' - g*\|_2 . \tag{1}$$

Given the expression of $d(g', g*)$, the loss function L^{Dist} of $Dist_\phi(c, g)$ is therefore represented as:

$$L_\phi^{Dist} = \|Dist_\phi(c, g) - d(g', g*)\|_2 . \tag{2}$$

In this work, $Dist_\phi(c, g)$ is implemented as an two-layer fully-connected (FC) neural network. The parameters ϕ are iteratively updated such that the loss function L_ϕ^{Dist} of $Dist_\phi(c, g)$ is minimized.

2.3 Goal Searching Process (GSP)

In this section, we present the details of goal searching process (GSP) in the instruction-goal mapping module. As the goal space $\mathcal{G}$ we consider in this paper is continuous and enumeration is impractical, GSP embraces a sampling-based searching algorithm that seeks to find a satisfactory pair of mean and variance (μ_g, σ_g) to sample a normal distribution of $\hat{g}$ within a given amount of time T_S. The sampling-based searching algorithm adopted in this paper is CEM, which casts the searching problem into an optimization problem. It repeats the sampling procedure until either the convergence condition or T_S is met. In each iteration, (μ_g, σ_g) is updated toward the direction that decreases $Dist_\phi(c, g)$, which serves as the metric function for evaluating the distances between c and the sampled $\hat{g}$'s.

3 Experimental Results

In this section, we present the experimental results and discuss their implications. We start by a brief introduction to our experimental setup. Next, we compare the performance and the learning

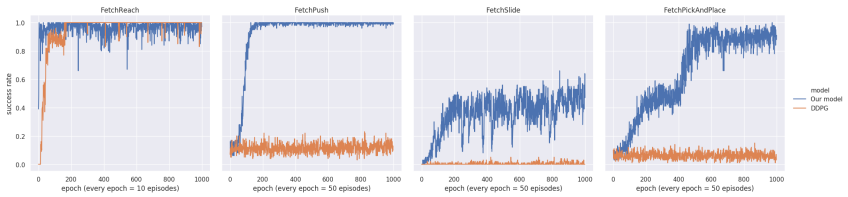

Figure 2: Comparison of our approach and the RL-based baseline.

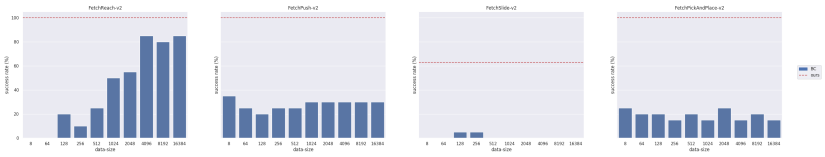

Figure 3: Comparison of our approach and the IL-based baseline.

efficiency of the proposed framework against a number of baseline methods. Finally, we validate the adaptability of our modular framework by replacing the input instruction set and robot action space to different ones, and demonstrate the adaptation efficiency of our framework over the baselines.

3.1 Experimental Setup

In this section, we introduce the environments and tasks, as well as the baselines used for comparison.

3.1.1 Environments and Tasks

We evaluate our framework on a number of robotic arm manipulation tasks via OpenAI Gym (Brockman et al., 2016) environments simulated by the MuJoCo (Todorov et al., 2012) physics engine. We use the Fetch robotic arm (Plappert et al., 2018) for the arm manipulation tasks, which include *FetchReach*, *FetchPush*, *FetchPickAndPlace*, and *FetchSlide*. The agent takes as inputs the positions and velocities of a gripper and a target *goal*. It then infers the gripper's action in 3-dimensional space to achieve the *goal*. The detailed description of the above tasks is specified in (Plappert et al., 2018).

3.1.2 Baselines

We adopt both RL- and IL-based approaches as the baseline methods. They are described as follows.
RL-based. We adopt DDPG as our RL-based baseline method, as it enables our agents to be trained in continuous action spaces. The RL-based agents receive a pair of instruction and observation (c, o) from the environment at each timestep, and are trained to reach the *goal* specified by c.
IL-based. In the IL-based baseline, we adapt the behavior cloning (BC) approach (Bain & Sammut, 1999) which learns to infer the demonstrated action given an observation o. The demonstration data used for training is collected by a pre-trained DDPG-HER agent, and contain tuples of (c, o, a).

Similar to the RL-based baseline, our proposed framework utilizes training data acquired from interactions between the agent and the environment (i.e., for training the conditional policy $\pi(a|o, g)$). Moreover, both the proposed framework and IL- based baseline employ training data that is demonstrated by an expert (i.e., the demonstration data is utilized for training the distance function $Dist_\phi$.).

3.2 Comparison of Data Efficiency in the Default Environment

In order to compare the data-efficiency of our method against that of the baselines, we evaluate their performances on the tasks described in Section 3.1.1 with a fixed instruction set and action space. In each task, the instruction sets employed for the training phase and the evaluation phase are the same. Similarly, the action spaces in both phases are consistent. Fig. 2 plots the learning curves of the models in each tasks. The x-axis of Fig. 2 indicates the number of training episodes. On the other hand, the x-axis of Fig. 3 represents the number of state-action pairs used by the IL-baseline. Note that the red dashed-lines in Fig. 3 correspond to the performance of our approach, and are obtained from 80 episodes of training. From Fig. 2, the results point out that learning the instruction-to-action mapping in an end-to-end fashion as the the RL-based baseline is more complex than our modular approach. Moreover, the results in Fig. 3 indicate that our approach delivers better performance in most of the tasks. Figs. 2 and 3 show that our method is able to yield same or superior performance comparing to the RL- and IL-based baselines, and is more data efficient and effective than them.

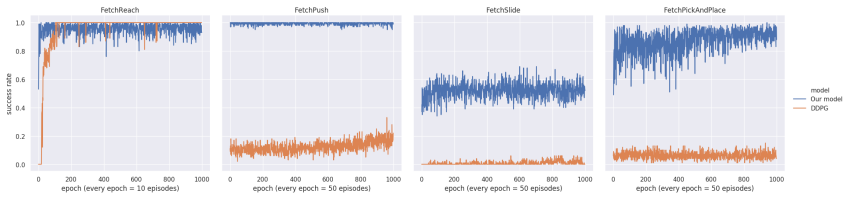

Figure 4: Comparison of adaptability with the RL-based baseline on a novel instruction set.

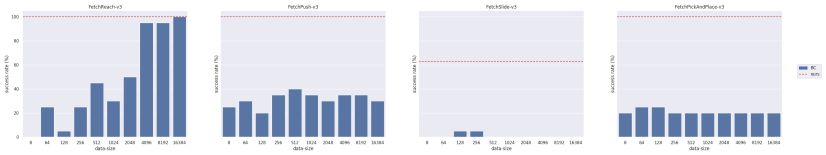

Figure 5: Comparison of adaptability with the IL-based baseline on a novel instruction set.

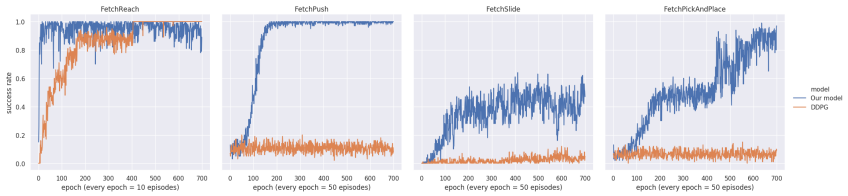

Figure 6: Comparison of adaptability with the RL-based baseline on a novel action space.

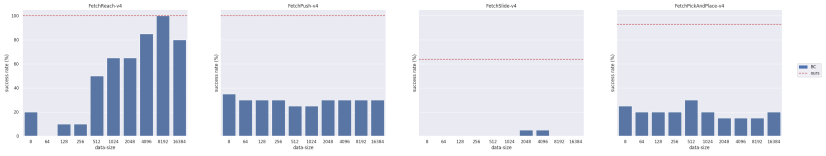

Figure 7: Comparison of adaptability with the IL-based baseline on a novel action space.

3.3 Comparison of Adaptability with a Different Instruction Set

In order to demonstrate the adaptability of our approach, we replace the original instruction sets with another one. In this experiment, we re-train the instruction-goal mapping module with new human demonstrations, while the entire models of the baseline approaches are re-trained.The results are plotted in Figs. 4 and 5. It is observed that the RL-based baseline in all the tasks and the IL-based baseline in most of the tasks with the novel instruction set obtain no significant improvement comparing to our approach. These results also reveal that compared to our modularized method, the end-to-end training approaches encounter difficulties in adapting their models to an unfamiliar instruction set. In Fig. 4, it cost the RL-based baseline at least twice more effort to reach the same performance as our approach in the simplest task *FetchReach*. On the other hand, in Fig. 5, the IL-based baseline reaches approximately comparable performance as ours with 16,384 demonstrations, while our proposed approach requires merely eight demonstrations.

3.4 Comparison of Adaptability with a Different Action Space

We further perform experiment for comparing the adaptability of our framework against the baseline approaches on a different action space. The results are plotted in Figs. 6 and 7. In this experiment, we only retrain the goal-policy mapping module of our framework. In Fig. 6, only the RL-based baseline in *FetchReach* is capable of reaching the same performance of our framework. However, it takes five times longer than ours, revealing the inefficiency of the RL-based baseline when adapting to an environment with an unfamiliar action space. On the other hand, the IL-based baseline is able to obtain the same performance as our proposed method in *FetchReach*. However, the IL-based baseline still necessitates 8,192 demonstrations while our approach only requires eight demonstrations. According to the experiments, our proposed approach achieves higher mean performances than all of the baselines, which validate the adaptability and data efficiency of the proposed modular approach.

4 CONCLUSIONS

We presented a modular framework for separating the instruction-to-action mapping procedure into two distinct stages. The first stage maps an input instruction to a *goal*, while the second stage maps the *goal* to a fitting policy selected from a set of robot policies. We implemented the above two stages as the instruction-goal mapping module and the goal-policy mapping module, respectively, and employed a distance function for evaluating the affinity between the instruction and the *goal*. We embraced the concept of metric learning in the development of our distance function, and used it to serve as a non-linear regressor to evaluate whether a given instruction is close to a *goal* or not. During the evaluation phase, we adopted CEM in our searching procedure. Our experimental results demonstrated that the proposed framework is able to learn an effective instruction-to-action mapping procedure in the OpenAI Gym robotic arm manipulation tasks, and is superior to the baseline methods in terms of efficiency. We further validated that the proposed framework is able to adapt to new instruction spaces and new robot action spaces much faster than the baseline methods.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 5048–5058, Dec. 2017.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, May 2009.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Proc. Machine Intelligence*, pp. 103–129, Jul. 1999.

Satchuthananthavale RK Branavan, Harr Chen, Luke S Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proc. the Joint Conf. of the ACL and the Int. Joint Conf. Natural Language Processing of the AFNLP*, pp. 82–90, Aug. 2009.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv:1606.01540*, 2016.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *Proc. the Thirty-Second AAAI Conf. Artificial Intelligence (AAAI-18)*, pp. 2819–2826, Feb. 2018.

David L. Chen and Raymond J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proc. the Twenty-Fifth AAAI Conf. Artificial Intelligence (AAAI-11)*, pp. 859–865, Aug. 2011.

David Silver *et al.* Deterministic policy gradient algorithms. In *Proc. Int. Conf. Machine Learning (ICML)*, pp. 387–395, Jun. 2014.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3D world. *arXiv:1706.06551*, Jun. 2017.

Cynthia Matuszek, Dieter Fox, and Karl Koscher. Following directions using statistical machine translation. In *Proc. ACM/IEEE Int. Conf. Human-Robot Interaction (HRI)*, pp. 251–258, Mar. 2010.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to parse natural language commands to a robot control system. In *Proc. Int. Symp. Experimental Robotics (ISER)*, pp. 403–415, Jan. 2013.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proc. the Thirtieth AAAI Conf. Artificial Intelligence (AAAI-16)*, pp. 2772–2778, Feb. 2016.

Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1004–1015, Sep. 2017.

Matthias Plappert et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv:1802.09464*, Mar. 2018.

Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

Jesse Thomason, Jivko Sinapov, and Raymond Mooney. Guiding interaction behaviors for multimodal grounded language learning. In *Proc. the First Workshop on Language Grounding for Robotics*, pp. 20–24, Aug. 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 5026–5033, Oct. 2012.

Terry Winograd. Understanding natural language. *Cognitive Psychology*, 3(1):1–191, Jan. 1972.