

Meta-Reinforcement Learning via Language Instructions

Zhenshan Bing, Alexander Koch, Xiangtong Yao, Kai Huang, Alois Knoll

Abstract—Although deep reinforcement learning has recently been very successful at learning complex behaviors, it requires a tremendous amount of data to learn a task. One of the fundamental reasons causing this limitation lies in the nature of the trial-and-error learning paradigm of reinforcement learning, where the agent communicates with the environment and progresses in the learning only relying on the reward signal. This is implicit and rather insufficient to learn a task well. On the contrary, humans are usually taught new skills via natural language instructions. Utilizing language instructions for robotic motion control to improve the adaptability is a recently emerged topic and challenging. In this paper, we present a meta-RL algorithm that addresses the challenge of learning skills with language instructions in multiple manipulation tasks. On the one hand, our algorithm utilizes the language instructions to shape its interpretation of the task, on the other hand, it still learns to solve task in a trial-and-error process. We evaluate our algorithm on the robotic manipulation benchmark (Meta-World) and it significantly outperforms state-of-the-art methods in terms of training and testing task success rates. Codes are available at <https://tumi6robot.wixsite.com/million>.

I. INTRODUCTION

In recent years, deep reinforcement learning (RL) has been applied very successfully to hard control tasks like playing video games [1]–[4], acquiring locomotion skills [5]–[7] and, robotic manipulation tasks [8]–[10]. However, learning these tasks often requires enormous amounts of environment interactions, which makes it impractical for many applications. For example, learning to manipulate a Rubik’s cube for a robotic hand, OpenAI reported a cumulative experience of 14,000 years for simulated interactions [10]. On the contrary, humans are able to manipulate the cube nearly instantaneously, as they have learned how to manipulate objects in general beforehand.

Meta-reinforcement learning (meta-RL) aims to design an efficient reinforcement learning algorithm to mimic the human learning ability that learns new tasks quickly [11]–[13]. Meta-RL algorithms achieve this by conditioning the policy on past experience and inferring the task information based on the received rewards [14]. Unfortunately, meta-RL algorithms perform poorly on diverse sets of tasks [15], since they solely rely on rewards to communicate the task to the agent, which is especially problematic when the rewards are sparse or indistinguishable among similar tasks. Therefore, providing additional information about the task to the agent offers a promising way to help the learning of new tasks.

Natural language provides a rich and intuitive way for humans and robots to interact with each other, due to the

Z. Bing, A. Koch, X. Yao, and A. Knoll are with the Department of Informatics, Technical University of Munich, Germany. E-mail: {bing, yaox, knoll}@in.tum.de

K. Huang is with the School of Data and Computer Science, Sun Yat-sen University, China. Email: huangk36@mail.sysu.edu.cn

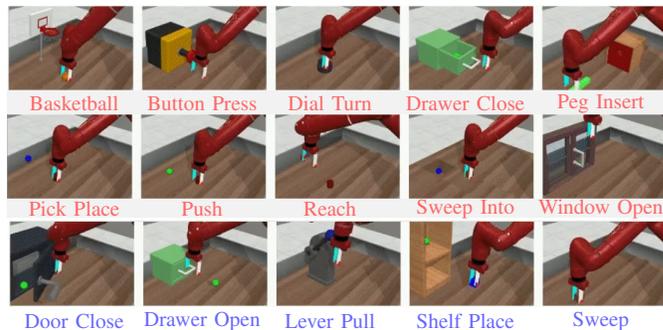


Fig. 1. A visualization of the ML10 benchmark from Meta-World. The first two rows show the training tasks and the last row shows the testing tasks. The figure is adapted from [15].

possibility of referring to abstract concepts. When a human worker is given a new task, they are usually told what to do by language, which specifies the task goal or the required skill. Therefore, the worker will not have to try every possible action sequence to figure out the goal, but purposefully aim at solving the specified task. Although language is the most intuitive way for humans to understand tasks, the topic of controlling a robot using language instructions is rather new and poorly understood.

With the fast development of algorithms in natural language processing, more and more studies that attempt to control robots via language instructions are beginning to emerge. Shao et al. proposed an imitation learning algorithm Concept2Robot [16], which aims to enable the robot to learn manipulation skills from language instructions and visual appearances of the task in two stages. In the first stage, Concept2Robot uses a video-based action classifier to generate a prediction score of the corresponding target task, which is served as a proxy reward to train the single-task policy. In the second stage, a multi-task policy is trained through imitation learning to imitate all the single-task policies. Stepputtis et al. [17] introduced an imitation learning model that directly maps labeled language instructions and visual observations to manipulation skills. Brucker et al. [18] proposed a flexible language based interface for human-robot collaboration, which allows a user to reshape existing trajectories for an autonomous agent. On the basis of imitating a large number of existing trajectories, the agent can generalize and adapt to new trajectories guided by the language. Lynch et al. [19] invented another algorithm that learns from existing expert demonstrations and adapt to solve tasks via multi-modal information to create the goal, such as languages or images. Clearly, we can see most existing algorithms learn language-conditioned skills via the concept of imitation learning, where

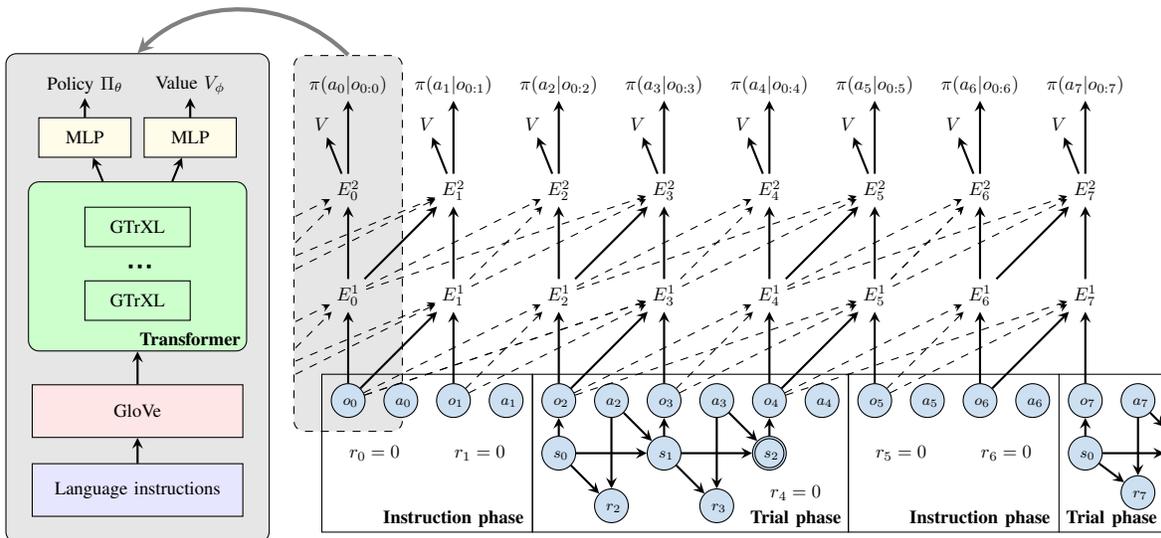


Fig. 2. Overview of our algorithm. Actions a_t are sampled from the distribution $\pi(a_t|o_{0:t})$. The dotted lines indicate how the memory segment in the GTrXL influences the hidden states. The memory segment before the first observation of the episode is initialized as a sequence of zero vectors. States with a double circle are terminal states. In our experiments we use a larger size of the GTrXL such that the agent can still use the observations from the first instruction phase to compute the last few actions of an episode.

large numbers of expert trajectories are required. This once again highly involves hand-crafted or engineered data and lacks the advantage of the trial-and-error learning paradigm, with which the agent can explore and learn the task by itself.

To this end, we establish a meta-RL algorithm that addresses the challenge of learning skills with language instructions in multiple manipulation tasks. We introduce the **Meta reINforcement Learning algorithm using Language INstruction (MILLION)**, which mimics the human-like learning manner and greatly improves the asymptotic performance in the challenging benchmark Meta-World. We base our method on three concepts.

- First, we propose a meta-RL learning paradigm that contains an instruction phase and a trial phase. In the instruction phase, the language description of the task is given to the agent, so that it can understand the goal of the task. In the trial phase, with the stored task information, the agent can explore and attempt to solve the task as standard reinforcement learning.
- Second, we build the architecture of our algorithm via three functional modules. The language instruction is encoded by a pre-trained language module and then taken as an input for a transformer module, where the information is stored and processed. The on-policy RL algorithm V-MPO is used to update the policy network and the value network.
- Experiment results demonstrate that MILLION significantly outperforms state-of-the-art algorithms on the challenging robotic manipulation benchmark (Meta-World [15], Figure 1), in terms of training and testing success rate. Previous works only achieve less than 50% success rate on the training tasks and less than 40% on the testing tasks, while MILLION achieves almost perfect performance on the training tasks and can solve

about half of the testing tasks.

II. METHODOLOGY

In this work, our goal is to propose a method that can provide the task information to the agent via instructions and learns to solve the task using trial-and-error RL algorithms. First, our policy network should be able to accept free-form language instructions of tasks as the input. Second, our method should use such instructions to communicate to the agent about what the task entails, instead of using extensive numbers of expert trajectories as other imitating learning based methods. Third, our method should enable the agent to successfully master diverse skills across broad tasks during training and adapt to unseen tasks during testing.

A. Overview

The architecture of MILLION is shown in Figure 2 and briefly explained as follows.

- First, an episode starts with an instruction phase, during which the language instructions are encoded as the observation using the pre-trained language model GloVe [20] and fed into the transformer module. The action generated by the policy network and the reward collected from the environment are simply ignored, since there is no interaction during the instruction phase.
- Second, after the instruction phase, a trial phase is started, during which the agent interacts with the environment by following the task’s Markov decision process (MDP). If the agent solves the task successfully, the environment will be reset and another trial phase starts. In the case of an unsuccessful trial, another instruction phase will start right after the trial phase, which resembles a real world scenario where a human operator might try a slightly different instruction to communicate the task to

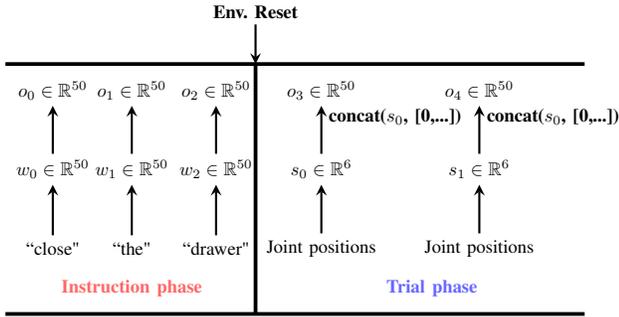


Fig. 3. Example of a language instruction during the instruction phase.

the agent. The whole episode will be terminated after a fixed number of trial phases.

- Finally, a new task is sampled and the same procedure will be executed.

B. Language Instruction Phase

We consider the problem of learning an instruction-conditioned control policy $\pi(a|s, \mathcal{I}(\tau))$, where \mathcal{I} represents language instructions about task τ . a is the selected action conditioned on the observation s . The instructions should be encoded into a sequence of vectors $[w_1, w_2, \dots, w_n] = \mathcal{I}(\tau)$, where $w_i \in \mathbb{R}^n, n \in \mathbb{N}$. We assume two phases in one training episode, namely, the **instruction phase** during which the task information is provided to the agent and the **trial phase** during which the agent interacts with the environment. The additional task information $\mathcal{I}(\tau)$ is only given to the agent in instruction phases, which can be expressed as

$$\begin{cases} \mathcal{I}(\tau) = [w_1, w_2, \dots, w_n]_{1 \times n} & , \text{ if } \mathbf{instruction\ phase} \\ \mathcal{I}(\tau) = [0, \dots, 0]_{1 \times n} & , \text{ if } \mathbf{trial\ phase} \end{cases} \quad (1)$$

During the instruction phase, the agent receives the encoded vectors in sequence and does not interact with the environment, thus the actions generated by the policy and the environment rewards are ignored. While in the trial phase, the agent does not receive new instructions, but interacts with the environment via the actions and rewards, therefore, $\mathcal{I}(\tau)$ is set as zero.

We provide free-form language instructions as the source of instructions, e.g., “open the drawer” for the drawer opening task, and “press the button” for the button pressing task. For every task, we create a set of language instructions l with similar key words. Some examples of the language instructions that we use for the ML10 benchmark are listed in Table I. At the beginning of the instruction phase, a new language instruction will be sampled for the current task τ . To capture the information represented in the natural-language command, we first use the GloVe algorithm [21] to convert the language instruction l into a sequence of fixed size vector $W = [w_0, \dots, w_T] = \mathcal{I}(l)$ with $w_i \in \mathbb{R}^{50}$, encoding up to T words with their respective 50-dimensional word embedding. This means that, at time step t of the instruction phase, the observation will be w_t . After T time steps the trial phase will start. An example of the instruction phase is visualized

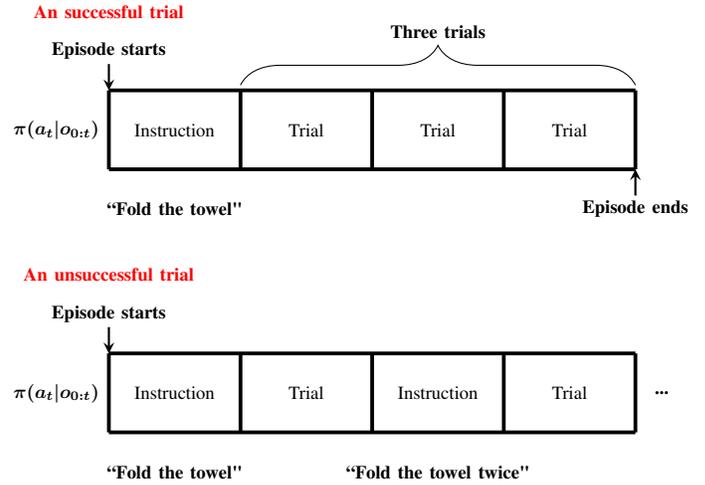


Fig. 4. The visualization of the phase sequence. Each episode starts with the instruction phase and follows with the trial phase. In case a trial is not able to solve the task, a new instruction phase will be added to enhance the understanding of the task.

in Figure 3. It should be noted that, to make the observations have the same length between the instruction phase and trial phase, a vector of zero is concatenated to the joint positions.

C. Trial Phase

The trial phase is defined as steps of environmental interactions between two resets of the environment by following the task’s MDP. The action policy Π and value policy V are updated by maximizing the accumulated rewards in the trial phase. The reset of the environment can be triggered by two conditions, namely, reaching a terminal state or reaching the maximum time-steps. As illustrated in Figure 4, we start each episode with an instruction phase and end the episode after three trial phases. In the event of a successful trial in which the agent solves the task, we continue the training with a new trial phase. In the event of an unsuccessful trial phase, we continue the training with a new instruction phase in which a similar language instruction is given to the agent. Following the same procedure, one trial phase will be initiated after each instruction phase.

TABLE I
EXAMPLES OF LANGUAGE INSTRUCTIONS FOR ML10

Task	Language Instructions
reach	reach to goal_pos, reach goal_pos
push	push goal_pos, push to goal_pos push object to goal_pos
pick-place	pick and place at goal_pos pick object and place at goal_pos
door-open	pull goal_pos, open door, pull to goal_pos
drawer-open	pull goal_pos, pull to goal_pos pull back to goal_pos
drawer-close	push goal_pos, push to goal_pos push forward to goal_pos
button-press-topdown	push object down to goal_pos, press button press down, press button down

Algorithm 1 MILLION

```

1: policy  $\pi_\theta(a|s)$ , state-value function  $V_\phi^\pi(s)$ 
2: initialize FIFO buffer  $\tilde{B}$  with capacity  $b * T_{target}$ 
3: while not converged do
4:   Update  $\pi_{\theta_{old}} \leftarrow \pi_\theta$ 
5:   for learning step  $l = 1..T_{target}$  do
6:     for trajectory number  $i = 1..b$  do
7:       Select instruction  $I(\tau)$  for random task  $\tau$ 
8:       Encode  $I(\tau)$  in language phase
9:       Do MDPs in trial phase with  $\pi_{\theta_{old}}(a|s, I(\tau))$ 
       to generate trajectory  $\Omega_\tau$ , and add  $\Omega_\tau$  to  $\tilde{B}$ 
10:       $B_{batch} = \text{Sample } b \text{ trajectories from } \tilde{B}$ 
11:      Reward normalize  $B_{batch}$ 
12:      Compute loss  $\mathcal{L}(\phi, \theta, \eta, \alpha_\mu, \alpha_\Sigma)$  from  $B_{batch}$ 
13:      Update  $\phi, \theta, \eta, \alpha_\mu, \alpha_\Sigma$  with gradient step

```

D. Reward Normalization

In multi-task RL or meta RL, one policy is trained to solve multiple tasks, from which the rewards typically have different magnitudes, for instance, in Meta-World (version 1), the task *press-button-v1* has a reward varying from 0 to 10,000 while *put-on-shelf-v1* has a reward varying from 0 to 10. This makes the learning extremely difficult and inefficient. A well-used solution is to clip the reward to a specified range.

Preserving outputs precisely, while adaptively rescaling targets (Pop-Art) [22] can be used to normalize the learning targets for the value function for every task individually. Inspired by Pop-Art, we also update the value function of our network as follows. The value function is used to predict the reward return G_t and is approximated as

$$f_{\theta, \sigma, \mu, w, b}(x) = \sigma(W h_{\theta/W, b} + b) + \mu, \quad (2)$$

where h is the neural network with the weights θ . W and b are parameters to normalize the prediction of the network. μ and σ are used to track the mean and standard deviation of the returns G_t . Then, μ and σ are updated as

$$\begin{cases} \mu_t = (1 - \beta)\mu_{t-1} + \beta G_t \\ \sigma_t = \sqrt{\nu_t - \mu_t^2} \\ \nu_t = (1 - \beta)\nu_{t-1} + \beta(G_t)^2 \end{cases} \quad (3)$$

where β is a training hyper-parameter. To keep the learning stationary, we update W and b as

$$\begin{cases} W_t = \frac{\sigma_{t-1}}{\sigma_t} W_{t-1} \\ b_t = \frac{\sigma_t b_{t-1} + \mu_{t-1} - \mu_t}{\sigma_t} \end{cases} \quad (4)$$

E. V-MPO with Improved Sample Efficiency

The policy is trained using the on-policy algorithm Maximum a Posteriori Policy Optimization (V-MPO). V-MPO is very sample inefficient. It requires a lot of environment interactions during training. We improve the sample efficiency by modifying the V-MPO algorithm slightly to reuse sampled environment interactions more often. The original V-MPO algorithm uses every environment trajectory only for one gradient update. We change this by keeping a small FIFO

buffer with the last $T_{target} \times b$ trajectories, where b is the batch size for the gradient updates. Then we randomly sample batches from this buffer for gradient updates.

The overall MILLION algorithm is given in Algorithm 1.

III. EXPERIMENTS

In this section, we evaluate the performance of our method on the well-known Meta-World benchmark that consists of 50 complex manipulation tasks. First, we apply MILLION to the ML10 benchmark to compare the performance against state-of-the-art meta-RL algorithms in terms of training and testing success rate. Second, we provide an ablation study on ML10 to validate the proposed concepts. Last, we conduct experiments on the most challenging benchmark ML45 to show its broad effectiveness and generalization capability.

A. Meta-World Benchmark

Meta-World [15] is a collection of 50 diverse robotic manipulation tasks built on the MuJoCo physics simulator [23]. It contains two widely-used benchmarks, namely, ML10 and ML45. The ML10 contains a subset of the ML45 training tasks, which are split into 10 training tasks and 5 test tasks, and the ML45 consists of 45 training tasks and 5 test tasks. Most tasks contain some kind of object that should be manipulated with the robot arm and adopt the control strategy:

- The action space \mathcal{A} contains the desired 3D Cartesian positions of the end-effector and a normalized control command for the gripper.
- The state space \mathcal{S} contains the 3D Cartesian positions of the end-effector, the positions of the manipulable objects, and the goal position. The state space is always nine dimensional.
- A success metric function is provided for each task, which defines the competition condition of the corresponding task.
- For each task, a well-shaped reward function is provided with a similar structure across all tasks, which makes the tasks individually solvable for recent RL algorithms.

We make two additional changes to the Meta-World benchmark to reduce the training time. First, inspired by [24], we repeat actions twice during the trial phase to reduce the trial length across all the tasks, which enables a shorter sequence length for the transformer model, and therefore reduces the computation requirements significantly. It should be noted that the reported number of environment steps in our results corresponds to the number of observations the agent has seen. Second, we add a scalar to the observations during the trial phase, which indicates the remaining time in the trial. This helps the agent to learn a better value function, because the Meta-World environments have a time dependent termination condition [25]. The time observation is computed as $\frac{\text{steps in one trial}}{\text{maximum steps per trial}}$. During the instruction phase, a zero value is concatenated to the observation instead.

There are two versions of Meta-World. Note that, in the first version of Meta-World, three tasks had to be removed from the benchmark, because the scripted policies provided by Meta-World did not work well to solve the tasks. This

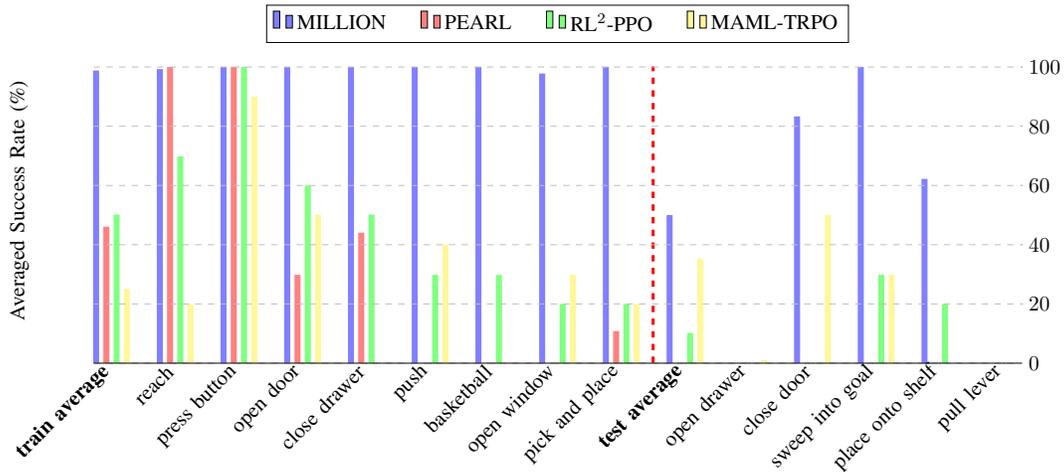


Fig. 5. Maximum per-task success rates on ML10 V1. MILLION shows the highest performance on the training tasks (98.8%) and the test tasks (50%).

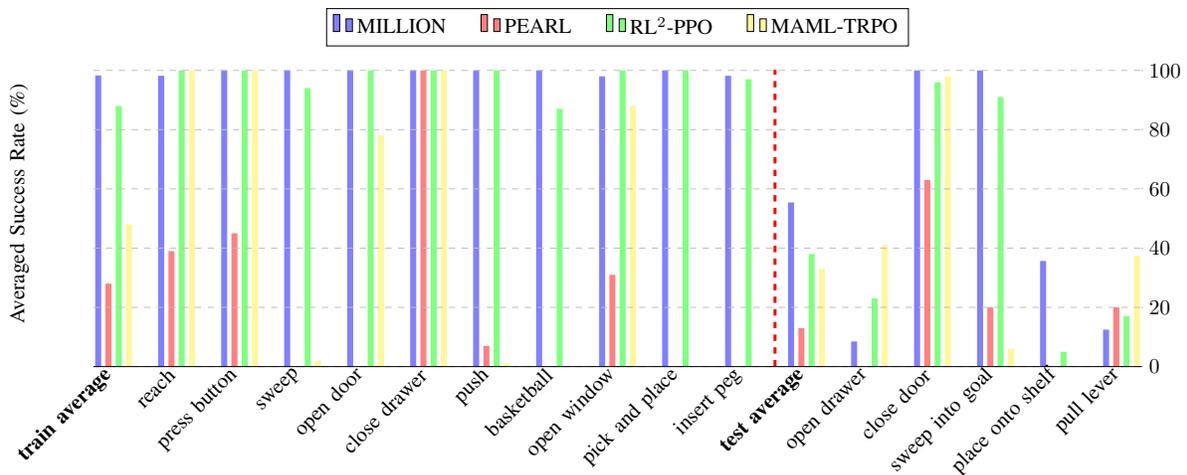


Fig. 6. Maximum per-task success rates on ML10 V2. MILLION shows the highest performance on the training tasks (98.3%) and the test tasks (55.4%).

includes peg-insert-side-v1, lever-pull-v1 and bin-picking-v1. Another task, sweep-v1, had to be removed because the reward function did not encourage the agent to solve the task. But for the second version of Meta-World, we keep all the tasks accessible for training and testing.

B. ML10 Benchmark

We first tested our method on the ML10 benchmark to show its performance when the agent receives a language instruction instead of only observing the reward signal. The language instructions are short sentences that describe the goal of the task. For each task in ML10, we designed multiple simple language instructions.

According to the reported results from [15], we listed the averaged success rates of state-of-the-art meta-RL algorithms in Table II, which includes MAML [26], RL² [27], PEARL [28], and our method MILLION. Detailed performance for each task in ML10 is visualized in Figure 5 and 6. It can be observed that, in both versions of Meta-World, MILLION achieves success rates of almost 100% on the training tasks, which significantly outperforms state-of-the-art methods. It

TABLE II
AVERAGE SUCCESS RATES OVER ALL TASKS FOR ML10 AND ML45.

Methods	ML10		ML45	
	Training	Testing	Training	Testing
MAML	25%	36%	21%	24%
RL ²	50%	10%	43%	20%
PEARL	43%	0%	11%	30%
MILLION	99%	50%	95%	48%

TABLE III
COMPARISON AMONG MILLION VARIANTS IN ML10 V1.

Variants	Meta-training	Meta-test
MILLION	0.99	0.50
without Pop-Art	0.41	0.30
without instructions	0.71	0.29
with Full Time Obs	0.83	0.40

demonstrates the advantage of providing the agent with the task instructions instead of only rewards. For meta-testing, MILLION has a success rate of around 50%, which also performs better than other methods (See Figure 7.).

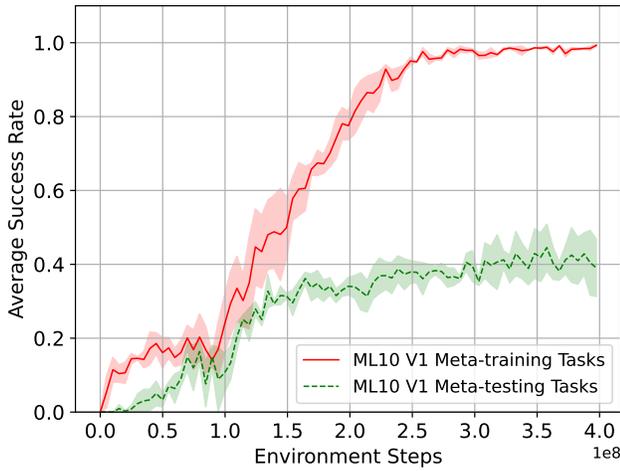


Fig. 7. Meta-World ML10 V1 training progress with language instructions. The shaded regions indicate one standard deviation over three training runs. The result of ML10 V2 benchmark can be found on the website¹.

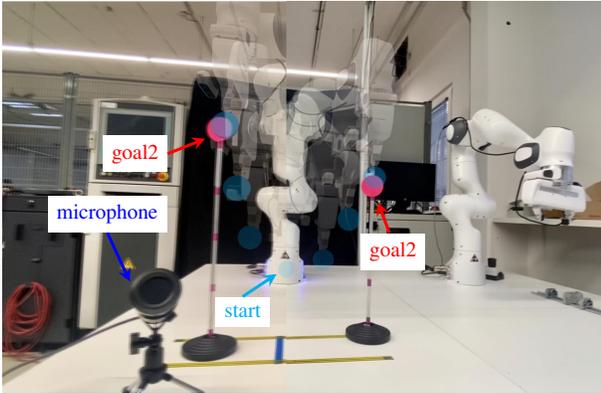


Fig. 8. We take the reaching task as one example to show that MILLION can be successfully used in the real world. The task are specified by the language instruction through the microphone. More demonstrations can be found on the project webpage¹.

To emphasize the importance of the proposed concepts, we provide ablation studies on the ML10 benchmark, shown in Table III. First, we demonstrate the importance of the Pop-Art reward normalization. This normalizes the rewards for every task individually. The results demonstrate that Pop-Art is very important for our algorithm. Without this, the agent only learns to solve less than half of the training tasks. Second, we also examine the performance of a variant that only observes rewards without instructions, which means an episode consists only of three trials and no instruction phase. The rewards are simply concatenated to the observations to serve as a potential information source of the task. This is similar to many other recent context-based meta-RL algorithms [27], [29]. This variant can learn 70% of the training tasks but adapts to the testing tasks poorly. Another ablation is to use a different time observation. Our algorithm observes the remaining time in the current trial. Here we evaluate our algorithm when it observes the remaining time in the full episode. This was originally proposed by [25]. The results show that this variant learns the training tasks slightly worse than MILLION. Our hypothesis

¹<https://tumi6robot.wixsite.com/million>

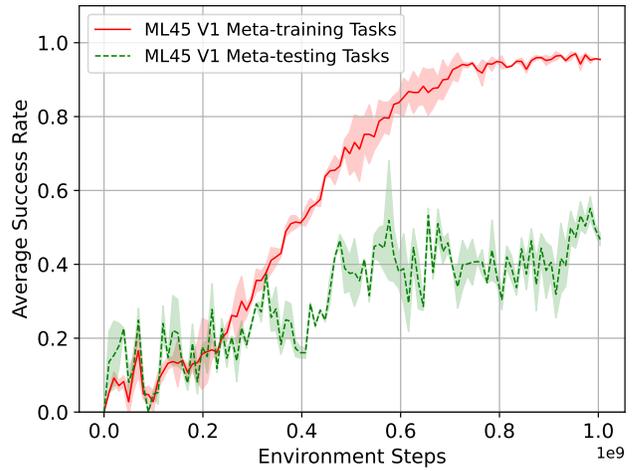


Fig. 9. Average trial success rate on ML45 training and test tasks. The policy is evaluated every 5 million environment steps and averaged over 5 consecutive evaluations.

is that the discount factor causes the value function during a trial to be relatively independent of the next trial rewards. This means that the remaining time is more important for the value function than the remaining time in the episode.

We also successfully transfer the learned policy from simulation to the real world. Due to the page limit, we only show the snapshot of MILLION solving the reach-v1 task in the real world (See Figure 8). More demonstrations of manipulation tasks from ML10 can be found on the webpage¹.

C. ML45 Benchmark

To test ML 45, we use the same hyperparameters and the same number of trials as for the ML10 benchmark. However, we train the agent for over 1 billion time steps instead of just 400 million because the benchmark contains more diverse tasks. The results (see Figure 9) show that MILLION is able to learn almost all training tasks and about 48% of the test tasks, which indicates that our method has a stable performance on complex manipulation tasks scenarios. A detailed comparison between MILLION and state-of-the-art algorithms on ML45 is also listed in Table II. It demonstrates that our algorithm MILLION greatly outperforms other baselines in terms of success rate in both training and testing stages.

IV. CONCLUSION

In this paper, we showed that meta-reinforcement learning can be greatly improved by providing the agent with additional task information, such as language instructions, which are often much easier to provide than dense rewards. By encoding the language instructions into the observations, we designed a very simple and general algorithm. This eases the application of RL algorithms to be used for real-world robotic tasks. Furthermore, we demonstrated that our algorithm is able to solve a set of very diverse robotic manipulation tasks. In future, we plan to incorporate the language information as a feedback signal to further calibrate the behavior of the meta-RL agent, which can potentially advance our understanding on interactive intelligent robots in the future.

REFERENCES

- [1] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.*, “Grandmaster level in starcraft ii using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
- [2] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, “Dota 2 with large scale deep reinforcement learning,” *arXiv preprint arXiv:1912.06680*, 2019.
- [3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, “Mastering chess and shogi by self-play with a general reinforcement learning algorithm,” *arXiv preprint arXiv:1712.01815*, 2017.
- [5] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.
- [6] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, “Soft actor-critic algorithms and applications,” *arXiv preprint arXiv:1812.05905*, 2018.
- [7] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap, “Distributed distributional deterministic policy gradients,” *arXiv preprint arXiv:1804.08617*, 2018.
- [8] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.
- [9] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, “Hindsight experience replay,” *Advances in neural information processing systems*, vol. 30, 2017.
- [10] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, *et al.*, “Solving rubik’s cube with a robot hand,” *arXiv preprint arXiv:1910.07113*, 2019.
- [11] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, “RL²: Fast reinforcement learning via slow reinforcement learning,” 2017. [Online]. Available: <https://openreview.net/forum?id=HkLXCE9lx>
- [12] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, “Learning to reinforcement learn,” *arXiv preprint arXiv:1611.05763*, 2016.
- [13] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, “Reinforcement learning, fast and slow,” *Trends in cognitive sciences*, vol. 23, no. 5, pp. 408–422, 2019.
- [14] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, “Efficient off-policy meta-reinforcement learning via probabilistic context variables,” in *International conference on machine learning*. PMLR, 2019, pp. 5331–5340.
- [15] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning,” in *Conference on robot learning*. PMLR, 2020, pp. 1094–1100.
- [16] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg, “Concept2robot: Learning manipulation concepts from instructions and human demonstrations,” *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1419–1434, 2021.
- [17] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor, “Language-conditioned imitation learning for robot manipulation tasks,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 13 139–13 150. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/file/9909794d52985cbc5d95c26e31125d1a-Paper.pdf>
- [18] A. Buckler, L. Figueredo, S. Haddadin, A. Kapoor, S. Ma, and R. Bonatti, “Reshaping robot trajectories using natural language commands: A study of multi-modal data alignment using transformers,” *arXiv preprint arXiv:2203.13411*, 2022.
- [19] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, “Learning latent plans from play,” in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 1113–1132. [Online]. Available: <https://proceedings.mlr.press/v100/lynch20a.html>
- [20] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
- [21] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
- [22] H. P. van Hasselt, A. Guez, M. Hessel, V. Mnih, and D. Silver, “Learning values across many orders of magnitude,” *Advances in neural information processing systems*, vol. 29, 2016.
- [23] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.
- [24] M. G. Bellemare, J. Veness, and M. Bowling, “Investigating contingency awareness using atari 2600 games,” in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.
- [25] F. Pardo, A. Tavakoli, V. Levdivik, and P. Kormushev, “Time limits in reinforcement learning,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 4045–4054.
- [26] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.
- [27] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, “RL²: Fast reinforcement learning via slow reinforcement learning,” 2017. [Online]. Available: <https://openreview.net/forum?id=HkLXCE9lx>
- [28] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, “Efficient off-policy meta-reinforcement learning via probabilistic context variables,” in *International conference on machine learning*. PMLR, 2019, pp. 5331–5340.
- [29] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, “Learning to reinforcement learn,” *arXiv preprint arXiv:1611.05763*, 2016.