Abstract

Visual navigation is a task of training an embodied agent by intelligently navigating to a target object (e.g., television) using only visual observations. A key challenge for current deep reinforcement learning models lies in the requirements for a large amount of training data. It is exceedingly expensive to construct sufficient 3D synthetic environments annotated with the target object information. In this paper, we focus on visual navigation in the low-resource setting, where we have only a few training environments annotated with object information. We propose a novel unsupervised reinforcement learning approach to learn transferable meta-skills (e.g., bypass obstacles, go straight) from unannotated environments without any supervisory signals. The agent can then fast adapt to visual navigation through learning a high-level master policy to combine these meta-skills, when the visual-navigation-specified reward is provided. Evaluation in the AI2-THOR [16] environments shows that our method significantly outperforms the baseline by 53.34% relatively on SPL, and further qualitative analysis demonstrates that our method learns transferable motor primitives for visual navigation.

1. Introduction

Visual navigation is a task of training an embodied agent that can intelligently navigate to an instance of an object according to the natural-language name of the object. In addition to being a fundamental scientific goal in computer vision and artificial intelligence, navigation in a 3D environment is a crucial skill for the embodied agent. This task may benefit many practical applications where an embodied agent improves the quality of life and augments human capability, such as in-home robots, personal assistants, and hazard removal robots.

Recently, various deep reinforcement learning (RL) approaches [42, 24, 40, 39, 31, 44, 45, 11, 21, 46, 19] have been proposed to improve the navigation models. They formulate the problem as the Partially Observable Markov Decision Process (POMDP) and train end-to-end policy networks to map observations to actions. However, deep RL methods are usually data inefficient and require a large amount of training data. In order to train these deep models, we need to construct a sufficient number of 3D synthetic environments and annotate the object information, which is exceedingly expensive, time-consuming, and even infeasible in real-world applications. Furthermore, it is hard for the trained embodied agent to transfer to different environments.

It is worth noticing that when humans encounter a new task, they can quickly learn to solve it by transferring the meta-skills learned in a wide variety of tasks throughout their lives. This stands in stark contrast with the current deep reinforcement learning-based navigation methods, where the policy networks are learned from scratch. Instead, humans have an inherent ability to transfer knowledge across tasks and cross-utilize their knowledge, which offloads the burden of a large number of training samples.

Inspired by this fact, we seek the help of both meta-learning [26, 7] that learn quickly using a small amount of data and transfer learning [37, 41] that accelerate learning a new task through transferring knowledge from a related task that is already learned. In our work, we frame low-resource visual navigation as a meta-learning problem. At the meta-training phase, the environments are not annotated with object information, and we assume access to a set of tasks that we refer to as the meta-training tasks. From these tasks, the embodied agent (we call it as meta-learner) then learns a set of transferable sub-policies, each of which corresponds to a specific meta-skill (also called as motor primitives, e.g., bypass obstacles, go straight) by perform-
Unsupervised Reinforcement Learning

Unannotated Scenes for Meta-training

Unseen Scenes for Meta-testing

Curriculum-Based Adversarial Training

Rewards during Adversarial Training: Meta-learner vs Task Generator

Meta-learner
Task Generator

Task 1
Task 2
... Task n

Meta-learning

Transferable Meta-Skills

Unsupervised Reinforcement Learning

Visual Navigation

Teddy Bear

Fast-Adaptation

Task-Specific External Reward

Master Policy

Sub-Policies

θ∗={θ1,...,θk}

θ1 θ2 θ3 ... θk

Figure 1: Overview of our ULTRA framework. The blue part on the left is our adversarial training process, where the task generator automatically proposes a curriculum of increasingly challenging tasks, and the meta-learner learns to complete them. From these tasks, the meta-learner learns a set of transferable sub-policies. Then, on the right part, the meta-learner can fast adapt to visual navigation by just learning a new master policy, given the task-specific external reward. The θk is corresponding to the parameters of the k-th sub-policy.

Our experimental results show that our method significantly outperforms the baseline by 53.34% on SPL. More-
over, further ablation study demonstrates the effectiveness of the adversarial training process and the hierarchical policy. Additionally, by qualitatively visualizing the behavior of the sub-policies, we find that the sub-policies show consistent motor primitives.

In summary, our contributions are mainly four-fold:

- We propose a novel ULTRA framework to learn transferrable meta-skills via unsupervised reinforcement learning.

- The hierarchical policy of meta-learner separates the entire policy into the task-specific part and task-agnostic part, which reduces the probability of meta-overfitting and promises a faster convergence.

- Instead of manually designing tasks, we propose a novel curriculum-based adversarial training strategy, where the task generator automatically proposes increasingly difficult tasks to the meta-learner. Further, we define a diversity measure to encourage the task generator to generate more diverse tasks.

- We perform our experiments in low-resource setting, and experimental results show that our method significantly outperforms the baseline by 53.34\% relatively on SPL and requires only one-third number of iterations to converge, compared with the baseline.

2. Related Work

Visual Navigation. Traditional navigation methods [3, 5, 14, 17, 20, 36] typically employ geometric reasoning on a given occupancy map of the environment. They perform path planning [4, 13, 18] to decide which actions the robot performs. Recently, many deep reinforcement learning (DRL) approaches [42, 24, 31, 44, 45, 11, 21, 46] have been proposed. They formulate the problem as the partially Observable Markov Decision Process (POMDP) and end-to-end learn policies network for visual navigation. While these methods achieve great improvement, it is difficult to apply them to real-world situations since these DRL methods require a large number of training episodes and annotated environment information, which is time-consuming and exceedingly expensive. In our work, we focus on developing an unsupervised reinforcement learning method in the low-resource setting.

Meta-Learning. Meta-learning, also known as learning to learn, optimizes for the ability to learn new tasks quickly and efficiently, using experience from learning multiple tasks. There are three common types of methods: 1) metric-based methods [32, 34, 38] that learn an efficient distance metric; 2) memory-based methods [22, 25, 27, 30] that learn to store experience using external or internal memory; and 3) gradient-based methods [26, 7, 12, 29, 9] model parameters explicitly for fast learning. Our method relies on a gradient-based meta-learning algorithm called Reptile [26]. The Reptile algorithm is aimed to learn a good parameter initialization during the meta-training process, where a large number of related tasks are provided. Thus, in the meta-testing process, the model can achieve good performance on new tasks after only a few gradient updates. An important difference is that our method does not require a large number of hand-designed tasks at the meta-training stage. Instead, we propose a curriculum-based adversarial training process that automates the meta-training process without any supervision.

Intrinsic Motivation-Based Exploration. Intrinsic motivation or curiosity called by psychologists have been widely used to train an agent to explore the environment and create environment priors without external supervision. There are mainly two categories of intrinsic reward: 1) incentivize the agent to explore “novel” states [6, 10, 33]; and 2) incentivize the agent to perform actions that reduce its predictive uncertainty of the environment [28].

Sukhbaatar et al. [33] introduce an adversarial training approach to unsupervised exploration, where one model proposes tasks and the other learns to complete it. In their work, the model for completing the tasks shares the whole parameters during training, and use the parameters as initialization for the downstream task. However, our work differs as we treat the adversarial training process as a sequence of independent meta-training tasks, and each task holds independent task-specific parameters. Also, there is no communication between two agents, whereas, in our work, the generator sends the target observation to the meta-learner, which contains the task information.

Gupta et al. [10] propose an unsupervised meta-learning method based on a recently proposed unsupervised exploration technique [6]. They use the heuristic method to define intrinsic reward (i.e. random discriminator, entropy-based method), which automates the task generation process during meta-training. There is no distinct task definition, and such a heuristic method is inefficient as the environment of visual navigation is complex and diverse. Our work instead introduces a curriculum-based adversarial training, which is more interpretable and efficient.

3. Method

3.1. Overview

As mentioned above, we frame low-resource visual navigation as a meta-learning problem. At meta-training phrase, the environments are unannotated, and no hand-designed reward is provided. The agent learns transferable meta-skills via our ULTRA from these scenes. At meta-testing phrase, a few annotated environments and corresponding visual-navigation-specified rewards are provided, and the agent needs to fast adapt to visual navigation. In this section, we
mainly focus on how to learn transferable meta-skills during meta-training.

Our goal is to use unsupervised reinforcement learning to learn transferable meta-skills that can be utilized by the embodied agent to quickly master visual navigation in indoor 3D scenes. During the curriculum-based adversarial training process, the task generator automatically proposes a curriculum of tasks, and the meta-learner learns to complete these tasks. Specifically, the architecture of the meta-learner is the shared hierarchical policy. For each task generated by the task generator, the meta-learner first reinitialize the master policy and learns to combine the sub-policies to complete the task. After adapting the master policy to the new task, the meta-reinforcement learning algorithm is applied to optimize the sub-policies to excellent performance across tasks.

3.2. Curriculum-Based Adversarial Task Generation

In this setting, we have two agents: a task generator and a meta-learner. As shown in Figure 2, during each iteration, the task generator starts at the initial state $s_0$, performs a sequence of actions, and finally stops at state $s_T$. Then, it sends its egocentric observation at the final state $s_T$ to the meta-learner. Given the observation $o_T$ at final state $s_T$, the goal of the meta-learner is to reach $s_T$ from $s_0$, which we call as a task. We initialize the meta-learner at state $s_0$, let it learn on this task for multiple episodes, and compute the success rate $r$. After that, the task generator proposes a new task, and the meta-learner repeats the above process.

The above adversarial training process does not involve any manually-designed tasks. The task generator automatically generates a curriculum of tasks for the meta-learner to complete. As the tasks become more and more complicated, the meta-learner needs to learn transferable sub-policies that are corresponding to meaningful motor primitives, so that it can fast adapt to the new tasks by learning a reinitialized master policy to combine the sub-policies.

Our goal is to automatically generate a curriculum of diverse tasks, where we first start with an easy task and then gradually increase the task difficulty. The reward function of the task generator consists of three components: a final reward based on the success rate, an intermediate reward that penalizes the task generator for taking too many steps, and a diversity measure that measures the diversity of the tasks.

**Success Rate:** We use the success rate of the meta-learner after multiple episodes to measure the difficulty of the task and give the generator a final reward. The final reward is defined as:

$$R_f = k \ast (1 - r)$$

where $k$ is a scaling factor, and $r$ is the success rate.
**Step Efficiency:** At each timestep, the task generator will receive a negative constant intermediate reward. We penalize the task generator for taking too many steps, which encourages it to generate the easiest task that the meta-learner can not complete. In the first few iterations, the task generator can propose tasks by performing a small number of steps. Then as the capabilities of the meta-learner increase, more steps will be taken to generate more difficult tasks (qualitative examples in Figure 2).

**Task Diversity:** In order to explore wider state spaces for our meta-learner to build a better visual and physical understanding of the environment, we add an additional item in the task generator’s reward function to encourage it to generate more diverse tasks. Formally, let $\pi$ denote the current policy, and $\pi'$ denote a previous policy. The diversity measure $D$ can be written as:

$$D = \sum_{s_t \in \tau} \sum_{\pi' \in \Pi} D_{KL}(\pi' (\cdot | s_t)) || \pi (\cdot | s_t)) \quad (2)$$

where $\tau$ is the trajectory from the current episode. $\Pi$ is the set of prior policies. We save the previous policy corresponding to the last four episodes in the set $\Pi$. We use KL-divergence to measure the difference between the current policy and the previous policies. The task diversity is aimed to incentivize the task generator to generate more diverse tasks that cover a larger state space of the environment.

Formally, the task generator’s total reward $R_G$ can be written as:

$$R_G = k \ast (1-r) - \lambda * n + \eta * \sum_{s_t \in \tau} \sum_{\pi' \in \Pi} D_{KL}(\pi' (\cdot | s_t)) || \pi (\cdot | s_t)) \quad (3)$$

where $\lambda$ and $\eta$ are weight hyper-parameters, and $n$ is the number of actions that the task generator executes.

For meta-learner, we use the shared hierarchical policy. We train it using actor-critic methods with rewards function that incentivizes it to reach the target.

3.3. Shared Hierarchical Policy

The shared hierarchical policy decomposes long-term planning into two different time-scales. At master-timestep, the master policy chooses a specific sub-policy from a set of sub-policies and then gives control to the sub-policy. As in [9], the sub-policy executes fixed N timesteps primitive actions(e.g. MoveAhead, RotateLeft) before returning control back to the master policy.

Formally, let $\phi$ denote the parameters of the master policy, and $\theta = \{\theta_1, \theta_2, ..., \theta_K\}$ denote the parameters of the K sub-policies. $\phi$ is the task-specific parameters, that is learned from scratch for each task. $\theta$ is shared between all tasks and switched between by task-specific master policies. For each task generated by the task generator during the adversarial training process, $\phi$ is randomly initialized at first and then optimized to maximize the total reward over multiple episodes, given fixed shared parameters $\theta$.

After fine-tuning the task-specific parameters $\phi$ to the task (called warm-up period), we take a joint update period, where both $\theta$ and $\phi$ are updated. The task-specific $\phi$ is optimized towards the current task, but the shared $\theta$ is optimized to excelent performance across tasks using gradient-based meta-learning algorithms. The details are discussed in the Sec 3.4.

3.4. Meta-Reinforcement Learning on the Proposed Tasks

Inspired by meta-learning algorithms that leverage experience across many tasks to learn new tasks quickly and efficiently, our method automatically learns transferable meta-skills from a curriculum of tasks generated in the adversarial training process.

**Background on Gradient-Based Meta-Learning:** Our method is inspired by prior work on a first-order gradient-based meta-learning algorithm called Reptile [26]. The Reptile algorithm is aimed to learn the initialization of a neural network model, which can fast adapt to a new task. The Reptile algorithm repeatedly samples a task, training on

---

**Algorithm 1** Unsupervised Reinforcement Learning

1: randomly initialize $\theta$, $\phi$, $\mu$
2: $\Pi \leftarrow \emptyset$
3: while not converged do
4:     $s_0 \leftarrow c_i, start\_state$
5:     collect rollout $\tau^G_i(s_0, s_1, ..., s_T)$ using $\pi^G_{\mu}$
6:     $s^* \leftarrow s_T$
7:     $o^* \leftarrow o_T$
8:     set task $\tau_i = SetTask(s_0, s^*, o^*)$
9:     for $w = 0, 1, ... W$ (warmup period) do
10:        collect rollout $\tau^w_i$ using $\pi^w_{\phi_i, \theta}$
11:        $\phi_i \leftarrow \phi_i + \alpha \nabla_\phi J(\tau^w_i, \pi^w_{\phi_i, \theta})$
12:        $\theta \leftarrow \theta + \alpha \nabla_\theta J(\tau^w_i, \pi^w_{\phi_i, \theta})$
13:     end for
14:     $\theta \leftarrow \theta + \beta (\bar{\theta} - \theta)$
15:     Evaluate $R_G$ as Eq 3 and update $\pi^G_{\mu}$
16:     if len($\Pi$) == 4 then
17:         $\Pi.pop(0)$
18:         $\Pi.append(\mu)$
19:     end if
20: end while
it, and moving the initialization towards the trained weights on that task.

Formally, let $\theta$ denote the parameters of the network, $\tau$ denote a sampled task, corresponding to loss $L_\tau$, and $\hat{\theta}$ denote the updated parameters after $K$ steps of gradient descent on $L_\tau$. The update rule of the Reptile algorithm is as follows:

$$\theta \leftarrow \theta + \beta(\hat{\theta} - \theta) \quad (4)$$

where the $(\theta - \hat{\theta})$ can be treated as a gradient that can be plugged into an adaptive algorithm such as Adam [15].

If we define $K = 1$, then this algorithm corresponds to joint training on the expected loss $\mathbb{E}_\tau [L_\tau]$. However, we perform multiple gradient updates, such that the update includes important terms from second-and-higher derivatives of $L_\tau$. Hence, the Reptile converges to a solution that is very different from the joint training.

For Visual Navigation, our goal is for the agent to learn transferable meta-skills from the unsupervised adversarial training process. Therefore, we apply the Reptile algorithm to update the hierarchical policy of the meta-learner. Different from the original Reptile algorithm that computes second-and-higher derivatives to update the whole parameters, we just apply it to update the parameters of the sub-policies and fix them during the test. Also, we treat $(\theta - \hat{\theta})$ as a gradient and use SGD to update it.

Algorithms [1] details our ULTRA that consists of four phases. Firstly, the task generator proposes a task. Secondly, the meta-learner joins in a warm-up period to fine-tune the master policy. Thirdly, the meta-learner takes a joint update period where both the master policy and sub-policies are updated. Finally, the task generator is updated based on the success rate of the meta-learner and repeats the above procedure.

Formally, let $\pi^G_\mu$ denote the policy of the task generator parameterized by $\mu$, and $\pi^M_{\phi_i,\theta}$ denote the policy of the meta-learner parameterized by task-specific parameters $\phi_i$ and shared parameters $\theta = \{\theta_1, \theta_2, \ldots, \theta_K\}$. Firstly, we run the task generator and collect a trajectory $\tau^G_i(s_0, s_1, \ldots, s_T)$. We then set the task $\tau_i$ for the meta-learner by the initial state $s_0$, final state $s_T$, and the observation $o_T$ at the final state. Secondly, we initialize the meta-learner using the shared sub-policies and the random-initialized master policy. We then run a warmup period to fine-tune the master policy. More specifically, we run the meta-learner for $W$ episodes, and use the collected $W$ trajectories to update the master policy $\phi_i$ as follows:

$$\phi_i \leftarrow \phi_i + \alpha \nabla_{\phi_i} J(\tau^w_i, \pi^M_{\phi_i, \theta}) \quad (5)$$

where $J(\tau^w_i, \pi^M_{\phi_i, \theta})$ is the objective function of any gradient-based reinforcement learning that uses the $w$-th trajectory of task $\tau_i$ produced by policy $\pi^M_{\phi_i, \theta}$ to update the master policy $\phi_i$. In our work, we use Asynchronous Advantage Actor-Critic(A3C) [23][43].

During the warmup period, the parameters of the shared sub-policies $\theta$ are fixed. After fine-tuning the master policy, we enter the joint update period, where we run the hierarchical policy for $J$ episodes, and update both $\phi_i$ and $\theta$ as follows:

$$\phi_i \leftarrow \phi_i + \alpha \nabla_{\phi_i} J(\tau^j_i, \pi^M_{\phi_i, \theta}) \quad (6)$$

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\tau^j_i, \pi^M_{\phi_i, \theta}) \quad (7)$$

More specifically, we save the value of $\theta$ before the joint update period. After $J$ times iterations, we get the updated parameters $\hat{\theta}$, and then we compute the gradient $(\theta - \hat{\theta})$ and update the shared sub-policies $\theta$ using Reptile Algorithm. Finally, we compute the final reward of the task generator based on the success rate $r$, step efficiency, and the diversity.

4. Experiments

In our experiments, we aim to (1) evaluate whether the agent can quickly learn visual navigation by leveraging the transferable meta-skills, given only a few training data, (2) determine whether the ULTRA is efficient than other unsupervised RL-based methods [6][10][28], (3) determine whether the hierarchical policy promises a better transfer, and (4) gain insight into how our unsupervised ULTRA works.

4.1. Experimental Setup

After learning transferable meta-skills using ULTRA, the agent is required to master visual navigation in a small number of episodes. Visual navigation is a task of training an embodied agent that can intelligently navigate to a specific object chosen by natural language. We can formally define visual navigation in the context of Markov Decision Process (MDP) with the state space $S$, action space $A$, transition dynamics $T$, and reward function $R$. Let $E = \{e_1, e_2, \ldots, e_n\}$ denote a set of scenes, $T = \{t_1, t_2, \ldots, t_m\}$ denote a set of target objects, and the visual observation $s_t$ denote the state at $t$ timestep. An episode can be determined by a scene, a target object, and an initial state. The action set $A$ consists of six unique actions (e.g. MoveAhead, RotateLeft, RotateRight, LookDown, LookUp, Done). The embodied agent is required to figure out the desired action $a_t$ at each timestep using only the egocentric RGB images and the language semantics of the target object. If the agent navigates to a position close enough to the target within a certain number of steps, we consider this episode to be successful.

We evaluate our approach in AI2-THOR [16] simulated environment, which is a photo-realistic customizable environment for indoor scenes and contains 120 scenes covering four different room categories: kitchens, living rooms, bedrooms, and bathrooms. We use 60 scenes for meta-training, 20 scenes for meta-testing, 20 scenes for validation, and 20
Table 1: **Quantitative results.** We compare variations of our method and the baselines on testing data. Additionally, we report the results on trajectories where the optimal path length is at least 5 \( (L \geq 5) \). Our ULTRA significantly outperforms the baselines, especially on \( L \geq 5 \), indicating the superiority of our method on long-term planning.

| Method          | All Success | SPL       | \( L \geq 5 \) Success | SPL       |
|-----------------|-------------|-----------|--------------------------|-----------|
| Random          | 8.21        | 3.74      | 0.24                     | 0.09      |
| A3C (learn from scratch) | 19.20        | 7.48      | 9.43                     | 4.13      |
| DIAYN           | 17.23       | 6.30      | 8.72                     | 3.79      |
| Curiosity       | 21.07       | 8.51      | 10.31                    | 4.37      |
| Ours            | **27.74**   | **11.47** | **20.57**                | **8.04**  |
| ULTRA – hierarchical policy | 24.27        | 10.54     | 14.13                    | 5.61      |
| ULTRA – adversarial training   | 20.23       | 8.35      | 10.04                    | 4.33      |

**Figure 3:** **Learning curves.** We report the rewards averaged across 10 evaluation tasks during meta-testing. After learning meta-skills using unsupervised meta-reinforcement learning, our ULTRA can fast adapt to visual navigation significantly faster than A3C (learn from scratch) and other state-of-the-art unsupervised RL-based methods.

**Figure 4:** **Ablation study of the number of the sub-policies.** We provide the results when we use different number of sub-policies.

**4.2. Results**

We summarize the results of our ULTRA and the baselines in Table 1. Also, we report the rewards averaged across 10 evaluation tasks during meta-testing in Figure 3. From Figure 3, we observe that our approach can fast adapt to visual navigation, significantly outperforming all baselines not only in learning speed but also in performance.
We illustrate the trajectories of some sub-policies. Each row represents the same sub-policy initialized in different scenes, while each column represents different sub-policies in the same location. Our sub-policies show consistent behaviors, that are corresponding to some meta-skills (sub-policy1 always bypasses obstacles, sub-policy2 always turns right, and sub-policy3 always turns left).

The number of iterations required for convergence of our ULTRA is about one-third of the baselines. Furthermore, as shown in Table 1 our approach achieves the best success rate and SPL, especially when the trajectory length $L \geq 5$, indicating the superiority of our method on long-term planning.

While DIAYN can learn useful skills on a variety of simulated robotic tasks studied in [6], it does not perform well on visual navigation: it even learns slower than A3C (learn from scratch), which means that it can not learn useful skills from unsupervised exploring. Also, compared with A3C (learn from scratch), the curiosity method makes limited improvement. We argue that the reason for this phenomenon is due to the complexity and diversity of the visual navigation environment, whose state space is always larger than the previous tasks.

### 4.3. Ablation Study

**Effect of Individual Components:** We conduct an ablation study to illustrate the effect of the hierarchical policy and the adversarial training in Table 1. We start with the final URLTML model and remove the hierarchical policy and the adversarial training, respectively.

The variation of ours without hierarchical policy uses a typical LSTM-A3C policy that updates the entire network during adversarial meta-training. We fine-tune the learned LSTM-A3C policy on meta-testing data. Removing the hierarchical policy, we notice that the success rate drops 3.47 points, and the SPL drops 0.93 points, indicating that updating the entire policy on a few training samples of each meta-training tasks results in poor transferability than ULTRA on visual navigation. Thus, the hierarchical policy reduces the probability of meta-overfitting.

Furthermore, the results of the last row (sample random location as meta-training tasks during unsupervised reinforcement learning) validate the superiority of curriculum-based adversarial training.

**Ablation of the Number of Sub-Policies:** To explore the impact of different numbers of the sub-policies, we modify the number of sub-policies. As illustrated in Figure 5, the success rate and SPL keeps increasing when the number of sub-policies is increased from 4 to 7. When we continue to increase the number of sub-policies, not only does the success rate not improve significantly, but SPL decreases because too many sub-policies results in confusion. In order to guarantee the performance and reduce the computational complexity, we set the number of the sub-policies to 7.

### 4.4. Qualitative Analysis

**Visualization of the task generator:** For a more intuitive view of how our curriculum-based adversarial training works, we visualize three qualitative examples in Figure 2. In each scenario, the tasks are generated starting from the same location. We can see that the difficulty of the generated tasks, corresponding to the length of the generated trajectories, increases as the serial number of the tasks goes up. Also, we can see that the generated trajectories in each scenario are in different directions, indicating that our task generator proposes diverse meta-training tasks.

**Behavior of the Sub-Policy:** We execute sub-policies separately in different scenes to visualize the learned meta-skills. In Figure 5, trajectories shown in each row represent the same sub-policies initialized in different scenes, and trajectories shown in each column represent different sub-policies in the same location. As illustrated in Figure 5, the same sub-policy shows consistent behavior in different scenes. Sub-policy1 always bypasses obstacles and goes straight, sub-policy2 always turns right, and sub-policy3 always turns left. The consistency of the sub-policies demonstrates that our ULTRA has learned meaningful meta-skills.

### 5. Conclusions

In this paper, we introduce a novel Unsupervised reinforcement Learning of TRAnsferable meta-skills (ULTRA) framework that enables the agent to learn transferable meta-skills from the curriculum-based adversarial training process. Experiments show that our method out-
performs the baselines by a large margin. Moreover, our method converges faster than baselines. Additionally, we find that the sub-policies show consistent motor primitives (e.g., bypass obstacles, go straight), indicating that the agent learns meaningful meta-skills via unsupervised reinforcement learning.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. arXiv preprint arXiv:1807.06757, 2018.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[3] Michael Blösch, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Vision based mav navigation in unknown and unstructured environments. In 2010 IEEE International Conference on Robotics and Automation, pages 21–28. IEEE, 2010.

[4] John Canny. The complexity of robot motion planning. MIT press, 1988.

[5] Mark Cummins and Paul Newman. Probabilistic appearance based navigation and loop closing. In Proceedings 2007 IEEE International Conference on Robotics and Automation, pages 2042–2048. IEEE, 2007.

[6] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. arXiv preprint arXiv:1802.06070, 2018.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1126–1135. JMLR. org, 2017.

[8] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. arXiv preprint arXiv:1704.05012, 2017.

[9] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. arXiv preprint arXiv:1710.09767, 2017.

[10] Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. arXiv preprint arXiv:1806.04640, 2018.

[11] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation: Supplementary material. 2017.

[12] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In International Conference on Artificial Neural Networks, pages 87–94. Springer, 2001.

[13] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE transactions on Robotics and Automation, 12(4):566–580, 1996.

[14] Kiyosumi Kidono, Jun Miura, and Yoshiaki Shirai. Autonomous visual navigation of a mobile robot using a human-guided experience. Robotics and Autonomous Systems, 40(2-3):121–130, 2002.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[16] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. A2-THOR: An Interactive 3D Environment for Visual AI. arXiv, 2017.

[17] Kurt Konolige, Motilal Agrawal, Robert C Bolles, Cregg Cowan, Martin Fischer, and Brian Gerkey. Outdoor mapping and navigation using stereo vision. In Experimental Robotics, pages 179–190. Springer, 2008.

[18] Steven M LaValle. Planning algorithms. Cambridge university press, 2006.

[19] Juncheng Li, Siliang Tang, Fei Wu, and Yueting Zhuang. Walking with mind: Mental imagery enhanced embodied qa. In Proceedings of the 27th ACM International Conference on Multimedia, pages 1211–1219. ACM, 2019.

[20] Larry Matthis and STEVENA Shafer. Error modeling in stereo navigation. IEEE Journal on Robotics and Automation, 3(3):239–248, 1987.

[21] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673, 2016.

[22] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. arXiv preprint arXiv:1707.03141, 2017.

[23] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pages 1928–1937, 2016.

[24] Arsalan Mousavian, Alexander Toshev, Marek Fiser, Jana Kosecka, Ayzaan Wahid, and James Davidson. Visual representations for semantic target driven navigation. arXiv preprint arXiv:1805.06066, 2018.

[25] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 2554–2563. JMLR. org, 2017.

[26] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999, 2018.

[27] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In Advances in Neural Information Processing Systems, pages 721–731, 2018.

[28] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 16–17, 2017.
