Unsupervised Reinforcement Learning of Transferable Meta-Skills for Embodied Navigation

Juncheng Li[1] Xin Wang[2] Siliang Tang[1] * Haizhou Shi[1] Fei Wu[1]
Yueting Zhuang[1] William Yang Wang[2]
[1]Zhejiang University [2]University of California, Santa Barbara
{junchengli, siliang, shihaizhou, wufei, yzhuang}@zju.edu.cn , {xwang, william}@cs.ucsb.edu

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. Experimental results show 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 (DRL) approaches [44, 26, 42, 41, 33, 46, 47, 13, 23, 48, 21] have been proposed to improve the navigation models. However, they 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 [28, 9] that learn quickly using a small amount of data and transfer learning [39, 43] 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 performing a sequence of primitive actions. At the meta-testing phase, a few an-

[*] Siliang Tang is the correspondence author.

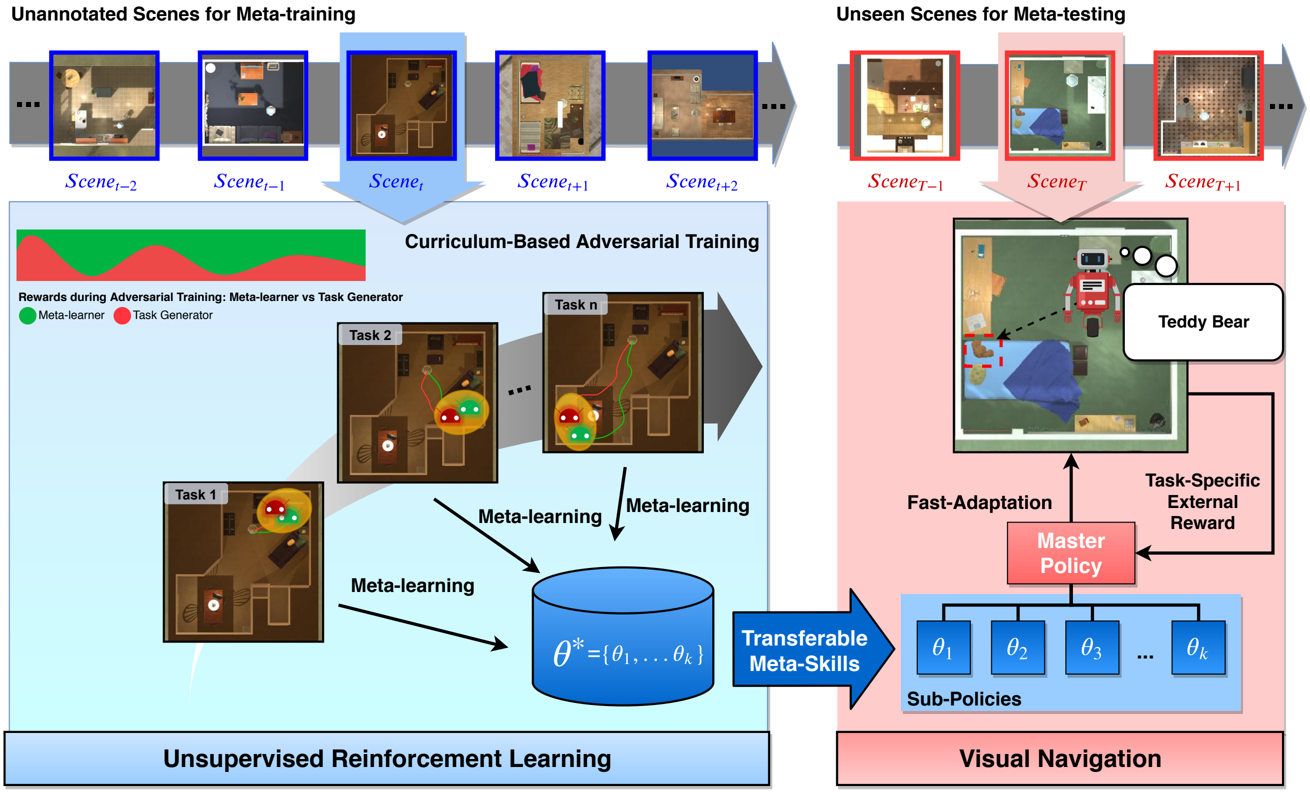

Figure 1: **Overview of our ULTRA framework.** The blue part on the left is our adversarial training process, where the task generator automatically proposes a curriculum of increasingly challenging tasks, and the meta-learner learns to complete them. From these tasks, the meta-learner learns a set of 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 θ_k is corresponding to the parameters of the k-th sub-policy.

notated environments with hand-specified rewards for visual navigation are provided. As illustrated in Figure 1, after learning transferable sub-policies from meta-training scenes, the agent is solely required to learn a new master policy to combine the sub-policies such that it can fast adapt to visual navigation. During meta-training, the master policy is task-specific, and the sub-policies are shared for all tasks across scenes. The master policy determines the execution order of the sub-policies and is optimized to fast adapt to each meta-training task. The sub-policies are optimized for performance across tasks using gradient-based meta-learning algorithms [28, 9]. The hierarchical architecture [11, 37, 3, 10] that separates the entire policy into the task-specific part and task-agnostic part can also avoid meta-overfitting: typical gradient-based meta-learning algorithms can easily result in overfitting since the entire network is updated on just a few samples.

However, typical meta-learning methods [28, 9] require a sufficient number of hand-designed tasks for meta-training, which is not practical for an embodied agent. In this paper, we then propose a novel unsupervised reinforcement learning approach that automatically generate a curriculum of tasks without manual task definition. In our Unsupervised reinforcement Learning of TRAnsferable meta-skills (ULTRA) framework, the agent can efficiently learn transferable meta-skills and thus fast adapt to the new task by leveraging the meta-skills when entering a new environment. The main body of the framework is what we call *the curriculum-based adversarial training process*, where one agent (*task generator*) generates a curriculum of tasks with increasing difficulty. The other agent (*meta-learner*) learns the meta-skills by accomplishing the generated tasks. After this unsupervised adversarial training process, the meta-learner can fast adapt to the new visual navigation task by just learning a new master policy to combine the learned meta-skills.

Our experimental results show that our method significantly outperform the baselines by a large margin, and further ablation study demonstrates the effectiveness of each component. Additionally, qualitative analysis demonstrates the consistent behavior of the sub-policies. In summary, our contributions are mainly four-fold:

- We propose a novel ULTRA framework to learn 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 [4, 6, 16, 18, 22, 38] typically employ geometric reasoning on a given occupancy map of the environment. They perform path planning [5, 15, 20]to decide which actions the robot performs. Recently, many deep reinforcement learning (DRL) approaches [44, 26, 33, 46, 47, 13, 23, 48] have been proposed. 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 [34, 36, 40] that learn an efficient distance metric; 2) *memory-based* methods [24, 27, 29, 32] that learn to store experience using external or internal memory; and 3) *gradient-based* methods [28, 9, 14, 31, 11] model parameters explicitly for fast learning. Our method relies on a gradient-based meta-learning algorithm called Reptile [28]. 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.

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 [8, 12, 35]; and 2) incentivize the agent to perform actions that reduce its predictive uncertainty of the environment [30].

Sukhbaatar *et al.* [35] 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.* [12] propose an unsupervised meta-learning method based on a recently proposed unsupervised exploration technique [8]. 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. Our work instead introduces an adversarial training strategy, which is more interpretable and efficient.

3. Method

In this section, we first define the meta-learning setting for visual navigation. We then describe our ULTRA framework. Finally, we discuss how to transfer the meta-skills to visual navigation.

3.1. Problem Setup

Our goal is to learn meta-skills in an unsupervised manner and then transfer the acquired meta-skills to new tasks (*i.e. visual navigation*). As illustrated in Figure 1, our approach has two stages: 1) In the meta-training stage, the agent learns transferable meta-skills via unsupervised reinforcement learning without human-specified reward functions. We use the curriculum-based adversarial training strategy to automatically generate a curriculum of meta-training tasks. 2) In the meta-testing stage, the agent is required to fast transfer to visual navigation task by utilizing the learned meta-skills. Training in this stage is fully supervised, but only a few training data is available.

Note that, the automatically generated meta-training tasks are different from visual navigation in the meta-testing stage. During meta-training, the learning target is to recover the x, y and viewing angle of the agent according to egocentric RGB observations and an image given by the task generator (called image-driven navigation). Different targets correspond to different tasks. While during meta-testing, the input to the agent is not an image, but a language command (*e.g., microwave*). The agent is required to under-

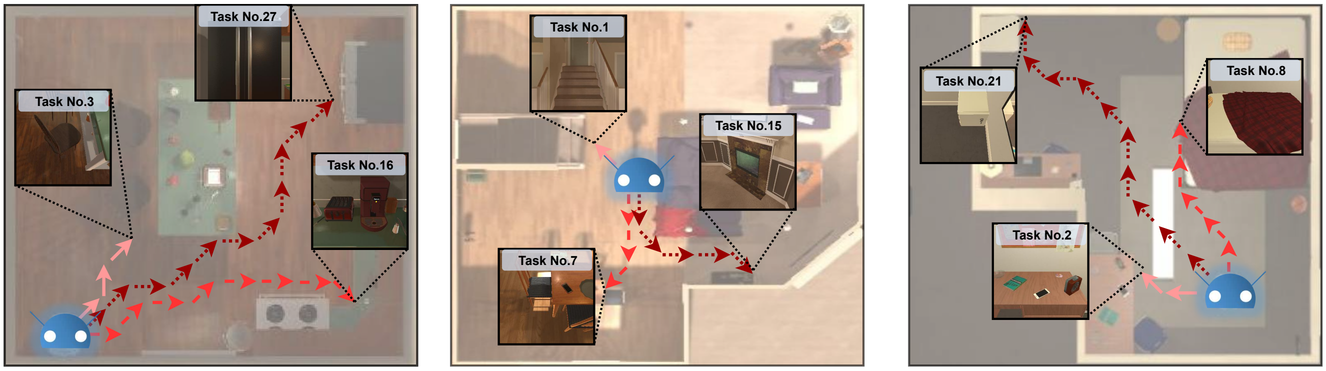

Figure 2: **Graphical illustration of the task generator.** The generator starts from the same location (denoted by the blue robot icon) and generates tasks for the meta-training. The level of difficulty (represented by the darkness of the path) increases along the training process.

stand various language commands and navigate to the objects specified by the commands in unseen scenes (called semantic visual navigation).

3.2. Overview

As described in Figure 1, our ULTRA framework mainly consists of three components: curriculum-based adversarial training strategy, shared hierarchical policy, and meta-reinforcement learning. 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, which contains a master policy and a set of sub-policies. At each master-timestep, the master policy first selects a sub-policy to be activated, and then the chosen sub-policy performs primitive actions. The master policy is learned from scratch for each task and encodes the task-specific information. The sub-policies are shared and encapsulate meta-skills that can be transferred across all tasks. 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.

3.3. Curriculum-Based Adversarial Training

In this setting, we have two agents: a task generator and a meta-learner. 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.

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:

$$R_f = k * (1 - r) \tag{1}$$

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 π denote the current policy, and π' denote a previous policy. The diversity measure D can be written as:

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

where τ is the trajectory from the current episode, Π is the set of prior polices. We save the previous policy corresponding to the last four episodes in the set Π. 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:

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

where λ and η 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[25] with rewards function that incentivizes it to reach the target.

3.4. 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 [11], the sub-policy executes fixed N timesteps primitive actions (*e.g. MoveAhead, RotateLeft*) before returning control back to the master policy.

Formally, let ϕ denote the parameters of the master policy, and $\theta = \{\theta_1, \theta_2, ..., \theta_K\}$ denote the parameters of the K sub-policies. ϕ is the task-specific parameters, that is learned from scratch for each task. θ 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, ϕ is randomly initialized at first and then optimized to maximize the total reward over multiple episodes, given fixed shared parameters θ.

After fine-tuning the task-specific parameters ϕ to the task (called warm-up period), we take a joint update period, where both θ and ϕ are updated. The task-specific ϕ is optimized towards the current task, but the shared θ is optimized to excellent performance across tasks using gradient-based meta-learning algorithms. The details are discussed in the Sec 3.5.

3.5. Meta-Reinforcement Learning on the Proposed Tasks

Inspired by meta-learning algorithms [28, 9, 14, 31, 11] that leverage experience across many tasks to learn new tasks quickly and efficiently, our method automatically learns meta-skills from a curriculum of meta-training tasks. **Background on Gradient-Based Meta-Learning:** Our method is inspired by prior work on a first-order gradient-based meta-learning algorithm called Reptile [28]. 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 it, and moving the initialization towards the trained weights on that task.

Algorithm 1 Unsupervised Reinforcement Learning

```
randomly initialize $\theta, \phi, \mu$
$\Pi \longleftarrow [\,]$
while not converged do
    $s_0 \longleftarrow e_i.start\_state$
    collect rollout $\tau_i^G(s_0, s_1, ..., s_T)$ using $\pi_\mu^G$
    $s^* \longleftarrow s_T$
    $o^* \longleftarrow o_T$
    set task $\tau_i = SetTask(s_0, s^*, o^*)$
    for $w = 0, 1, ...W$ (warmup period) do
        collect rollout $\tau_i^w$ using $\pi^M_{\phi_i,\theta}$
        $\phi_i \longleftarrow \phi_i + \alpha\nabla_\phi J(\tau_i^w, \pi^M_{\phi_i,\theta})$
    end for
    $\tilde{\theta} = \theta$
    for $j = 0, 1, ...J$ (joint update period) do
        collect rollout $\tau_i^j$ using $\pi^M_{\phi_i,\tilde{\theta}}$
        $\phi_i \longleftarrow \phi_i + \alpha\nabla_\phi J(\tau_i^j, \pi^M_{\phi_i,\tilde{\theta}})$
        $\tilde{\theta} \longleftarrow \tilde{\theta} + \alpha\nabla_\theta J(\tau_i^j, \pi^M_{\phi_i,\tilde{\theta}})$
    end for
    $\theta \longleftarrow \theta + \beta(\tilde{\theta} - \theta)$
    Evaluate $R_G$ as Eq 3 and update $\pi_\mu^G$
    if $len(\Pi) == 4$ then
        $\Pi.pop(0)$
    end if
    $\Pi.append(\mu)$
end while
```

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

$$\theta \longleftarrow \theta + \beta(\tilde{\theta} - \theta) \tag{4}$$

where the $(\theta - \tilde{\theta})$ can be treated as a gradient that includes important terms from second-and-higher derivatives of L_τ. 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 police 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 $(\theta - \tilde{\theta})$ 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 π^G_μ denote the policy of the task generator parameterized by μ, and $\pi^M_{\phi_i,\theta}$ denote the policy of the meta-learner parameterized by task-specific parameters ϕ_i and shared parameters $\theta = \{\theta_1, \theta_2, ..., \theta_K\}$. Firstly, we run the task generator and collect a trajectory $\tau_i^G(s_0, s_1, ..., s_T)$. We then set the task τ_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 ϕ_i as follows:

$$\phi_i \longleftarrow \phi_i + \alpha \nabla_\phi J(\tau_i^w, \pi^M_{\phi_i,\theta}) \tag{5}$$

where $J(\tau_i^w, \pi^M_{\phi_i,\theta})$ is the objective function of any gradient-based reinforcement learning that uses the *w-th* trajectory of task τ_i produced by policy $\pi^M_{\phi_i,\theta}$ to update the master policy ϕ_i. In our work, we use Asynchronous Advantage Actor-Critic(A3C) [25, 45].

During the warmup period, the parameters of the shared sub-policies θ 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 ϕ_i and θ as follows:

$$\phi_i \longleftarrow \phi_i + \alpha \nabla_\phi J(\tau_i^j, \pi^M_{\phi_i,\tilde{\theta}}) \tag{6}$$

$$\tilde{\theta} \longleftarrow \tilde{\theta} + \alpha \nabla_\theta J(\tau_i^j, \pi^M_{\phi_i,\tilde{\theta}}) \tag{7}$$

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

3.6. Transferring to Semantic Visual Navigation

In the meta-testing stage, we fix the learned sub-policies from meta-training process, and employ the Asynchronous Advantage Actor-Critic(A3C) [25, 45] to train a new master policy on a few new scenes. The inputs of the master policy are the egocentric observation of current state and the word embedding of the target object (e.g., *microwave*). In this stage, human-specified reward functions for visual navigation are available. If the agent reaches the target object within a certain number of steps, the agent receives a positive final reward. Also, it receives a negative intermediate reward at each step. Finally, we evaluate the performance on unseen scenes.

	All		$L \geq 5$	
	Success	SPL	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 - ULTRA	**27.74**	**11.47**	**20.57**	**8.04**

Table 1: **Quantitative results.** We compare our approach with 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.

4. Experiments

In our experiments, we aim to (1) evaluate whether the agent can quickly transfer to 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 [8, 12, 30], (3) determine whether the hierarchical policy promises a better transfer, and (4) gain insight into how our unsupervised ULTRA works.

4.1. Experimental Setup

We evaluate our approach in AI2-THOR [17] 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 choose 60 scenes for meta-training, and 60 scenes for meta-testing. For the 60 meta-testing scenes, we further divide them into three splits (*i.e.* 20 scenes for supervised training, 20 scenes for validation, and 20 scenes for testing). During meta-training, object information and hand-specified rewards for visual navigation are not accessible, and the agent performs unsupervised reinforcement learning to learn transferable meta-skills. During meta-testing, all models are fine-tuned or learned from scratch on the training set, and are finally evaluated on the testing set. we choose the same set of navigational target object classes as [44], and the training reward is specific since the human-annotated labels are available. The action set A consists of six unique actions (*e.g. MoveAhead, RotateLeft, RotateRight, LookDown, LookUp, Done*).

Task and Evaluation Metric: We use the averaged rewards on evaluation tasks during the training process to evaluate the learning speed, success rate to evaluate the navigation performance, and the success weighted by Path Length

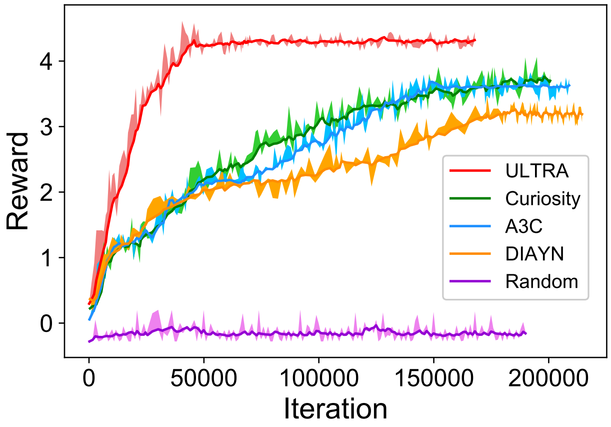

Figure 3: **Learning curves.** We report the rewards averaged across 10 evaluation tasks during meta-testing.

(SPL)[1] [1] to evaluate the navigation efficiency. As [44], we report the performance both on all trajectories and trajectories, where the optimal path length is at least 5 ($L \geq 5$).

Baselines: We compare our method with the following baselines: (1) **Random policy:** The agent randomly execute an action at each timestep; (2) **A3C (learn from scratch):** The architecture is the same as ours. However, there is no ULTRA process, and the whole hierarchical policy is directly learned from scratch in the meta-testing stage with visual-navigation-specified rewards.

We also compare to the state-of-the-art unsupervised RL-based methods: (3) **Curiosity:** [30] The agent learns skills motivated by a curiosity reward, which serves as an intrinsic reward and is the error in an agent's ability to predict the consequence of its own actions in a visual feature space learned by a self-supervised inverse dynamics model. (4) **DIAYN:** [8, 12] Diversity-driven methods hypothesize there is a latent variable (control different skills) behind the agent state distribution and maximize the mutual information between the latent variable and the agent state distribution. They then combine the unsupervised skill acquisition (via DIAYN) with MAML[9]. They train a discriminator to predict the latent variable from the observed state. As our ULTRA, DIAYN and Curiosity first conduct unsupervised reinforcement learning on scenes for meta-training and then are fine-tuned on training scenes for meta-testing with visual-navigation-specified rewards.

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. We observe that our approach can fast adapt to visual navigation, significantly outperforming all baselines not only in learning speed but also in performance. The number of iterations required for convergence of our ULTRA is about one-third of the baselines. Furthermore, 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. DIAYN breaks down for visual navigation, because the same state can be reached via different skills from different initial state, which causes the discriminator performs at chance. 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.

	All		$L \geq 5$	
	Success	SPL	Success	SPL
A3C (learn from scratch)	19.20	7.48	9.43	4.13
A3C + random generator	19.73	7.12	9.31	4.47
A3C + hand-crafted generator	20.57	8.04	10.26	4.28
ULTRA	**27.74**	**11.47**	**20.57**	**8.04**
– hierarchical policy	24.27	10.54	14.13	5.61
– meta-RL update	23.57	11.03	14.02	5.49
– adversarial training	20.23	8.35	10.04	4.33

Table 2: **Ablation results.** We compare variations of our method with the A3C baseline augmented with random generator and hand-crafted generator.

4.3. Ablation Study

Effect of Individual Components: We conduct an ablation study to illustrate the effect of each component in Table 2. We start with the final ULTRA model and remove the hierarchical policy, meta-RL algorithm, and the adversarial training, respectively. Furthermore, we augment the baseline with pre-training on meta-training scenes with a random generator and hand-crafted generator [19]. The augmented A3C baselines are first pre-trained on scenes for meta-training with a random generator or hand-crafted generator and then fine-tuned on training scenes for meta-testing. The random generator samples random location as the target, while the hand-crafted generator first samples the initial state closer to the target, and gradually increases the distance between the initial state and the target state. The augmented baselines regard different targets as different episodes under a unified task and employ A3C to learn a policy. The augmented baselines correspond to typical pre-training methods that pre-train on a source task (image-driven navigation) and transfer to the target task (semantic visual navigation) by fine-tuning the parameters.

Augmenting the baseline with pre-training on image-driven navigation proposed by the random generator or hand-crafted generator, we notice that there is no significant improvement and the performance is worse than Curiosity.

[1] The SPL is defined as $\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(p_i,l_i)}$, where N is the number of episodes, S_i is a binary indicator of success in episode i, l_i is the shortest path distance, and p_i is the path length.

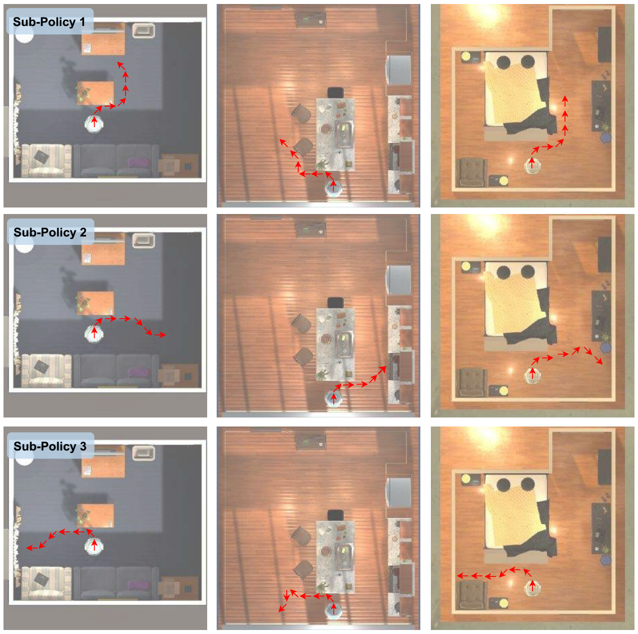

Figure 4: Visualization of the Sub-Policies.

The result reveals that image-driven navigation can not directly benefit semantic visual navigation.

The variation of ours without hierarchical policy uses a typical LSTM-A3C policy that updates the entire network during adversarial meta-training. 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. We then verify the strength of the meta-RL algorithm. As the pre-training baselines, we regard different targets as different episodes of the same task and update the parameters iteratively. Evidently, the meta-RL update improves upon the baseline by considering different targets as different meta-training tasks and updating the sub-policies by Reptile. 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: Figure 2 shows three qualitative examples. In each scenario, the tasks are generated starting from the same location. We can see that the difficulty of the 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.

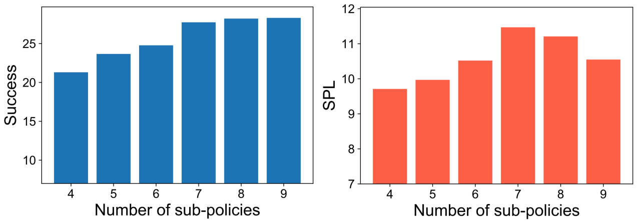

Figure 5: Ablation study of the number of the sub-policies.

Behavior of the Sub-Policy: We execute sub-policies separately in different scenes to visualize the learned meta-skills. In Figure 4, 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 4, 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 ULTRA framework that enables the agent to learn transferable meta-skills in an unsupervised manner. Experiments show that our method can fast transfer to semantic visual navigation and outperform the baselines by a large margin. Additionally, we find that the sub-policies show consistent motor primitives. The ULTRA framework provides a new perspective that fuses the meta-learning and transfer-learning in a more interpretable way and in the future we plan to transfer the meta-skills to other tasks (*i.e. Vision-and-Language Navigation [2], Embodied Question Answering [7] etc.*).

Acknowledgment

This work has been supported in part by National Key Research and Development Program of China (SQ2018AAA010010), NSFC (No.61751209, U1611461), Hikvision-Zhejiang University Joint Research Center, Zhejiang University-Tongdun Technology Joint Laboratory of Artificial Intelligence, Zhejiang University iFLYTEK Joint Research Center, Chinese Knowledge Center of Engineering Science and Technology (CKCEST), Engineering Research Center of Digital Library, Ministry of Education. The authors from UCSB are not supported by any of the projects above.

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] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

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

[4] 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.

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

[6] 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.

[7] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2054–2063, 2018.

[8] 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.

[9] 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.

[10] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

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

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

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

[14] 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.

[15] 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.

[16] 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.

[17] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

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

[19] Jonáš Kulhánek, Erik Derner, Tim de Bruin, and Robert Babuška. Vision-based navigation using deep reinforcement learning. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2019.

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

[21] 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.

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

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[32] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[33] Manolis Savva, Angel X Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931*, 2017.

[34] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[35] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

[36] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[37] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[38] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[39] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global, 2010.

[40] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[41] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019.

[42] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 37–53, 2018.

[43] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[44] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6750–6759, 2019.

[45] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.

[46] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.

[47] Haonan Yu, Xiaochen Lian, Haichao Zhang, and Wei Xu. Guided feature transformation (gft): A neural language grounding module for embodied agents. *arXiv preprint arXiv:1805.08329*, 2018.

[48] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017.