Learning Hierarchical Policies in Dynamic Environments

Vigneshram Krishnamoorthy
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
vigneshk@andrew.cmu.edu

Sumit Kumar
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
skumar2@andrew.cmu.edu

Suriya Narayanan Lakshmanan
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
snlakshm@andrew.cmu.edu

Abstract

We propose a general framework for solving sparse rewards or long horizon tasks in dynamic environments using a hierarchical approach combined with meta reinforcement learning. Our framework first learns useful skills and then utilizes these acquired skills for solving the hierarchical task. Using meta-learning, we learn a general representation of each skill across the distribution of environments achieving fast adaptation when fine-tuned with a few gradient updates. A high-level policy trained on top of these fine-tuned skills provides an efficient exploration strategy towards solving sparse reward tasks. Through experiments, we demonstrate the effectiveness of our proposed method on solving locomotion tasks in sparse reward dynamic environments.

1 Introduction

Reinforcement Learning (RL) has achieved many impressive results in recent years ranging from achieving superhuman performance at Atari games [23], game of Go [31] and learning advanced manipulation and locomotion skills [30, 21, 19]. However, the vast majority of the RL research is confined to simulation environments like OpenAI gym, Mujoco [33], Bullet, etc. where environmental attributes such as friction, viscosity, etc. and agent physique such as limb configurations are fixed. This limits the use of learned policies in real-world applications where the environment is dynamic and the agent needs to adapt to different conditions to accomplish the given task. Also, RL agents need a large amount of time and experience to learn new tasks from scratch and can be sample-inefficient, hence requiring the need to be able to formulate ways for generalization and fast adaptation to new tasks based on the agent's prior experience. For example, an agent trained with a specific geometry in a particular environment when deployed in real-life is bound to have mismatches in geometry and can be exposed to a different environment where fast adaptation is essential. Animals provide us great intuition for such fast adaptation to new tasks since they learn by trial and error on how to compensate for changes in their environment or their body as they grow.

Meta-learning addresses this problem by training models on a variety of tasks such that it can solve new tasks using only a small number of training samples from the current setting. Using standard RL frameworks, we experimentally validate that a pre-trained agent is unable to generalize when faced with an unseen task and performs poorly, motivating our meta-learning approach. We learn a generalized representation of the skill required to solve the task using meta-learning, which can perform well in environments that are dynamic and is immune to changes in the physical attributes of the agent. Furthermore, we use these skills to solve tasks with sparse rewards or long horizons where naive exploration strategies like ϵ-greedy or uniform Gaussian exploration noise perform poorly.

In this paper, we formulate a novel framework in which the RL agent can tackle variations in its physique, dynamic environments and is successful in solving a sparse-reward or long-horizon task. Our formulation brings together the domain of hierarchical reinforcement learning and meta reinforcement learning. We use hierarchical RL for making the framework sample efficient by learning a higher-level policy over the set of skills learnt using meta learning. We present results comparing the performance of a pre-trained RL agent to our meta RL agent in varying environmental conditions which demonstrates the effectiveness of our proposed approach. Using our approach, we also illustrate quantitatively and qualitatively, a notable improvement in performance in solving sparse-reward tasks in dynamic environments compared to a pre-trained agent.

2 Related Works

Solving tasks with sparse rewards or long horizons is still a big challenge in Deep RL community. Researchers have proposed two main strategies in this domain: The first strategy is to design a hierarchy over the actions [32, 24, 8]. Sutton et al. [32] proposed the *options* framework which involved abstractions over the space of actions. By combining low-level actions into high-level primitives or sub-policies, the search space can be reduced exponentially. The second strategy is to guide exploration by using intrinsic rewards [3, 18]. Although, domain knowledge is not required to compute these intrinsic rewards, these methods often require solving each task from scratch as the knowledge of solving one task is not directly transferable to others. As a result, while solving a collection of tasks, the overall sample complexity is high.

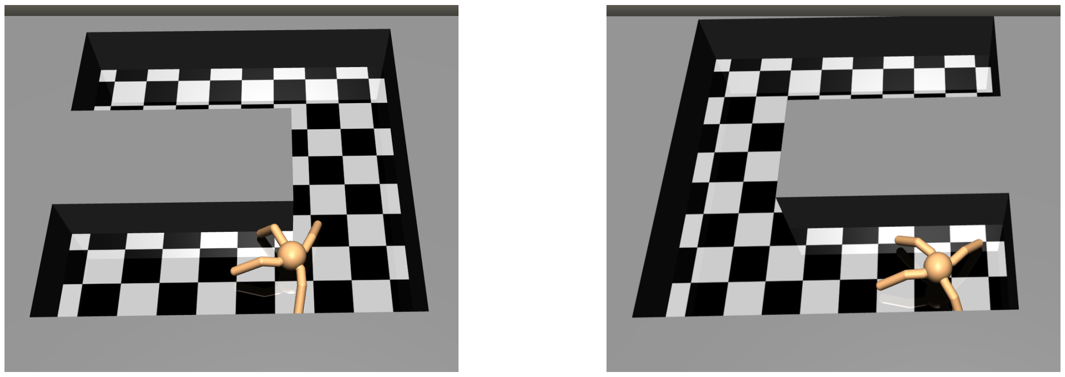
By using sub-policies or skills as high-level primitives, the search space can be reduced exponentially in HRL tasks. Prior works on learning skills have focused in discrete domains [5, 34] as well as in continuous action spaces [7, 27]. Bacon & Precup [2] proposed the Option-Critic architecture that can learn interpretable skills however whether these skills can be reused across complex tasks is still an open question. Heess et al. [15] proposed to learn a range of skills in a pre-training environment useful for downstream tasks. Their pre-training setup requires a set of goals to be specified. Similar to their work, Florensa et al. [13] proposed a general framework that first learns useful skills in a pre-training environment and then leverages the acquired skills for learning faster in downstream tasks. An agent learns useful skills with the help of a proxy reward signal designing which requires domain knowledge. Then, a high-level policy is trained on top of these skills providing a significant improvement of the exploration and tackling sparse rewards. Eysenbach et al. [11] proposed a soft actor-critic[14] based method to learn diverse skills without any extrinsic reward from the environment. However, all these methods have not been able to learn good sub-policies in high-dimensional continuous control environments like Mujoco Ant environment. As a result, they achieve poor performance on the corresponding hierarchical task [9]. In this work, we have focused on learning useful skills in a pre-training environment with a custom-designed reward function. These skills are then be used by a higher-level policy to solve the hierarchical task.

Meta reinforcement learning has emerged as a promising approach to enable an agent to learn and adapt quickly to a new task drawn from a distribution from only a few examples rather than learning every task from scratch. A popular approach to meta-learning is to train a meta-learner that learns to update the learner's model's parameters [4, 28]. This approach has been applied to learning to optimize deep networks [16, 20, 1]. Ravi et al. [25] learned both the weight initialization and the optimizer for few-shot image recognition. Another line of research trains memory-augmented networks on many tasks, where the recurrent learner is trained to adapt to new tasks. The applications of this approach can be seen in few-shot image recognition [26] as well as in learning "fast" RL agents [35, 10]. Mishra et al. [22] proposed a temporal convolution-based architecture which reportedly outperformed the previous memory-based approaches on a variety of RL tasks. Cully et. al [6] demonstrated how trial-and-error learning algorithms could allow robots to creatively discover compensatory behaviors using prior knowledge so as to adapt to injury. Houthooft et. al [17] proposed a meta-learning approach which evolves a differentiable loss function which optimizes its policy to minimize this loss parameterized via temporal convolutions over the agent's experience which enables fast task learning. Finn et al. [12] proposed a model-agnostic algorithm called MAML for training the model's initial parameters such that the model has maximal performance on a new task after updating its parameters through one or more gradient steps computed with a small amount of data from that new task. In this work, we have used MAML algorithm to learn good initializations for sub-policies that are later adapted to the actual task using only a few training examples.

3 Problem Statement

We define a finite-horizon discounted MDP by a tuple $M = (S, A, P, r, \rho_0, \gamma, H)$ where S is the state space, A is the action space, $P : S \times A \times S \to \mathbb{R}_+$ is the transition probability distribution, $r : S \times A \to \mathbb{R}$ is the reward function, $\rho_0 : S \to \mathbb{R}_+$ represents an initial state distribution, $\gamma \in (0, 1]$ is the discount factor and H is the horizon. The agents aims to optimize a stochastic policy $\pi_\theta : S \times A \to \mathbb{R}_+$ parameterized by θ in order to maximize the expected discounted return, $\eta(\pi_\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t)\right]$ where $\tau = (s_0, a_0, \cdots, s_H), s_0 \sim \rho_0(s_0), a_t \sim \pi_\theta(a_t|s_t)$ and $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$.

We consider the Locomotion + Maze hierarchical task with ant agent from rllab [10]. A **task** is represented as an MDP M where the aim of the agent is to reach the goal state in the given maze (see Fig. 1). The agent receives a reward of +1 on successfully completing the task and 0 otherwise. The following 3 entities vary across tasks: (1) the maze configuration: maze layout and obstacle locations, (2) the friction coefficients between surface and the agent, and (3) the agent's physical parameters namely torso and ankle sizes. $\mathcal{M}$ represents the collection of the all such MDPs M. Given this distribution $\mathcal{M}$, our objective is to minimize the total sample complexity required to solve the tasks. This is a challenging problem because the agent not only has to devise an efficient exploration strategy but also has to adapt to the dynamicity in the environment in order to solve the task.



(a) Maze 1 (b) Maze 2

Figure 1: Illustration of the sparse reward tasks considered in this work. Maze 2 is the reflection of Maze 1.

4 Methods

In this section, we describe our hierarchical formulation to solve a collection of structurally related tasks as explained in the previous section. In Sec. 4.1, we explain our methodology for learning skills or sub-policies in a pre-training environment. In Sec. 4.2, we describe our architecture of higher-level policy or master policy over the learned skills or sub-policies to solve the task.

4.1 Learning sub-policies

Given a distribution of tasks $\mathcal{M}$, we aim to learn a set of useful skills that will later be used as high-level primitives or macro-actions by a master policy. In order to solve the Locomotion + Maze task, the agent needs to learn to navigate efficiently in different directions without falling over. Since, any motion in $2D$ can be decomposed into movements along the four mutually orthogonal directions :left, right, up and down, we have used only 4 skills corresponding to these movements. These four skills can be thought of forming a set of basis skills from which complex locomotion skills can be derived. This skill-set may not be sufficiently powerful in certain environments like circular or spiral mazes where complex maneuvers are required, but for the environments considered in this work they perform well as demonstrated by our results.

An **environment configuration** describes the rllab environment, where the friction coefficients between the ant and the floor, torso size of the ant and the four ankle sizes of the ant are the parameters. We define a **sub-policy task** as learning to navigate in one of the four directions: left, right, up or down whilst minimizing drift in other directions in a given environment configuration. We aim to learn a general representation for these sub-policies that can quickly adapt to changes in the environment configuration. In order to achieve this, we formulate the sub-policy learning problem as a meta-learning problem aimed at learning a good initialization that can be fine-tuned to the specific task.

We use model agnostic meta-learning (MAML) [12] as our meta-learning framework. Being model-agnostic, MAML gives us the flexibility in the underlying model representation and policy optimization techniques. Moreover, its simplicity and few hyper-parameters makes it an ideal choice. Furthermore, Finn et al. [12] have demonstrated that their proposed method can scale to more complex deep RL problems like the high-dimensional locomotion tasks with the MuJoCo simulator [33].

For training the sub-policy tasks, we modified the reward structure of the rllab ant environment so as to give a positive reward proportional to the velocity of the agent in the desired direction and penalize the robot for drifting in the perpendicular direction. The custom-designed reward structure in our work is as follows:

$$\begin{aligned} r_{+x} &= v_x - 2|v_y| \\ r_{-x} &= -v_x - 2|v_y| \\ r_{+y} &= v_y - 2|v_x| \\ r_{-y} &= -v_y - 2|v_x| \end{aligned}$$

where r_{+x} is the reward function for the sub-policy task of navigating along the positive x-axis and similarly r_{-x}, r_{+y} and r_{-y}. v_x and v_y represents the agent's velocity along the x-axis and the y-axis respectively. We found that a factor of 2 was crucial to reduce the drifting of agent in other directions. We represent the skill corresponding to navigation along the $+x$ axis with ψ_{+x} and similarly ψ_{-x}, ψ_{+y} and ψ_{-y}.

Algorithm 1 Learning sub-policy initialization using MAML

```
Requires : $p(\mathcal{E}), \alpha, \beta$
Output   : Sub-policy $\psi$
Initialize $\psi$ randomly
while not done do
    Sample batch of environment configurations $\mathcal{E}_i \sim p(\mathcal{E})$
    for all $\mathcal{E}_i$ do
        Sample $K$ trajectories $\mathcal{D} = \{(x_1, a_1, ...x_H)\}$ using $f_\psi$ in $\mathcal{E}_i$
        Evaluate $\nabla_\psi \mathcal{L}_{\mathcal{E}_i}(f_\psi)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{E}_i}$
        $\psi_i^{'} \leftarrow \psi - \alpha \nabla_\psi \mathcal{L}(f_\psi)$
    end
    Sample $K^{'}$ new trajectories from each $\mathcal{E}_i$ using corresponding $f_{\psi_i^{'}}$
    $\psi \leftarrow \psi - \beta \sum_i \nabla_\psi \mathcal{L}(f_{\psi_i^{'}})$
end
```

Algorithm 1 describes our proposed skill learning method using MAML. Note that the same algorithm is executed four times with different reward structures to learn a good initialization for all four sub-policies or skills. During meta-training, a batch of environment configurations $\mathcal{E}_i$ is randomly sampled from the given distribution and the model is trained using K trajectories from each one of the sampled environment configurations. During the adaptation step, the updated model parameters

$\psi_i^{'}$ corresponding to each configuration $\mathcal{E}_i$ are computed from the configuration-specific loss $\mathcal{L}(f_\psi)$ by performing a few gradient updates for each $\mathcal{E}_i$.

During the meta-optimization step of updating ψ, we sample $K^{'}$ new trajectories for each of these configurations $\mathcal{E}_i$ and compute the loss on the network containing the adapted model parameters $\psi_i^{'}$. Then, we take the gradients of the above loss w.r.t the generalized model parameters ψ. Hence, the meta-optimization is performed over the model parameters ψ, while the objective is computed using the adapted model parameters $\psi_i^{'}$. α and β are the step-size parameters for the adaptation step and the meta-optimization step respectively. Since $\nabla_\psi \mathcal{L}(f_{\psi_i^{'}}) = \nabla_\psi \mathcal{L}(f_{\psi - \alpha \nabla_\psi \mathcal{L}(f_\psi)})$, this update requires the computation of a gradient through a gradient. The expected reward is not differentiable due to the unknown transition dynamics, hence, we use a policy gradient method namely PPO [29] to estimate the gradients for both the adaptation step and the meta-optimization step. The fine-tuning of the generalized model parameters by performing only a few gradient updates on the test environment configuration is termed as meta-testing. We also train a single NN model using PPO over the same task distribution which will be hereby referred to as the pre-trained network. Using experiments, we prove that the MAML model can perform fast adaptation to the new test environment, whereas the pre-trained model performs comparatively poorly under the same conditions.

4.2 Learning High-level policies

Given a set of K skills learned during the pre-training phase, we now describe how to use them as high-level primitives for solving a sparse reward task. Note that the skills learnt using MAML serve as good initialization on the training distribution and need to be updated further to the new task. For any given task $M \in \mathcal{M}$, we first fine-tune the skills ($\psi_{+x}, \psi_{-x}, \psi_{+y}$ and ψ_{-y}) to adapt to the current task using a few gradient steps with a small amount of training data from the current task. This meta-update is performed using the same reward structure used for learning the respective sub-policy. This is necessary as the reward signal from the environment is sparse and meta-update step requires dense reward signals.

The fine-tuned sub-policies are then frozen and a new Manager NN is trained on top of them. This master network selects on the K fine-tuned sub-policies. The agent then interacts with the environment using the selected sub-policy for h time-steps after which the master network makes another selection. Hence, from the perspective of the master policy network, h environment time steps corresponds to a single transition. The pseudo code of the training procedure of master policy is shown in Algorithm 2.

Algorithm 2 Learning master policy

```
Requires : Sub-policies $\Psi = \{\psi_{+x}, \psi_{-x}, \psi_{+y}, \psi_{-y}\}$, Task $M$
Output   : Master policy $\theta$
forall $\psi \in \Psi$ do
    Fine-tune $\psi$ to adapt to the task $M$
end
Initialize $\theta$
while not done do
    while Episode not terminated do
        Select sub-policy $\psi$ from the master policy $\theta$
        Execute $\psi$ for $h$ time-steps
    end
    Update $\theta$ to maximize the expected return from $\frac{1}{h}$ timescale viewpoint
end
```

5 Experiments

We used **rllab** framework for our carrying out our experiments. Specifically, we focused on Locomotion + Maze hierarchical task environment with ant agent. The rllab Ant is an 8-dof agent with four limbs and a spherical torso. The complex dynamics of the ant make it a difficult reinforcement learning environment. Florensa et al. [13] have attempted to solve the same task as ours using their proposed Stochastic Neural Network-based hierarchical framework but their trained ant is unstable while switching skills and topples over. As a result, their proposed framework is unable to solve the task. To eliminate this toppling problem, we reduced the gear setting of the ant resulting in a decreased max speed. This modification dramatically increased the stability of the ant by making it less susceptible to toppling.

The ant agent learns the set of skills by interacting with an empty environment with no mazes. The learning procedure is described in Section 4.1. During MAML training, the environment configuration vary only across episodes and is fixed for the entire episode. The range of distribution for the agent attributes such as torso and ankle sizes are from 75% of the original value to 125% of the original value (symmetric across the four ankles) during training. The friction coefficient varies from 0.1 to 0.8 along both the x and y axes.

We run a set of experiments to investigate the adaptation capability of the pre-trained and MAML sub-policies in a new test environment configuration. Using the trained MAML and pre-trained models, we perform meta-testing as described in Section 4.1 using only 4 gradient updates. The learning rate during meta-testing was set at 0.01 for the MAML and pre-trained policies.

After learning the skill-set Ψ, the agent learns a task-specific master policy θ by interacting with the maze environment with the help of its learned skills. In this task, apart from the joints positions and velocities, the agent also receives LIDAR-like sensor readings of the distance to walls and goals that are within a certain range. These extra sensor readings constitute the observation space for the master policy network. Our second set of experiments are concerned towards using these sub-policies towards solving the sparse-reward tasks and performing a comparative analysis between MAML-trained and pre-trained sub-policies.

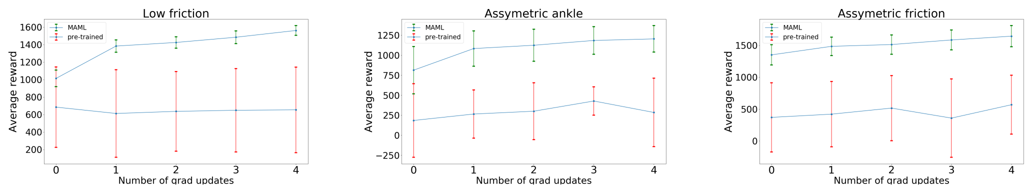
All the networks in our implementation are actor-critic networks with both the actor and the critic being 3 layer MLP with 64 hidden units in each layer. We found $h = 200$ works well in our application, i.e., the master policy selects a sub-policy after every 200 time-steps. The learning rate is set as $5e$-4 and the discount factor .995. We used the Adam optimizer for training the models. The update frequency for the network was set at 2 episodes while training the sub-policies and 4 episodes while training the master policy. We used PPO for training both the sub-policies and the master policy wtih the number of PPO epochs for each update set at 10.

6 Results

Table 1 illustrates the significant difference in performance between the pre-trained agent and the MAML agent at the sub-policy tasks of navigating in left, right, up or down directions. We present the results (in the same order as the records in Table 1) for the following test-set configurations: (1) low friction setting where the friction coefficients and the increased torso size are outside the training distribution, and the ankle sizes being the default values (2) asymmetric ankle configuration with the ankle sizes also outside the training distribution, (3) asymmetric friction configuration with an increased torso size with the ankles taking the default values. Figure 2 shows the training curve across number of gradient updates during meta-testing for each of the 3 configurations. From Figure 2 we can infer that the MAML initialization works better in all these configuration than the pre-trained initialization. More importantly, the MAML policy is able to adapt to the new,unseen configuration in just 4 updates, whereas the pre-trained fails to show significant improvement after the same number of gradient updates. During meta-testing, the variance in rewards also decreases for the MAML policy whereas the pre-trained model always has much high variance in rewards.

Ant Parameters			Rewards Pre-trained		Rewards MAML	
Friction (x,y)	Torso size (units)	Ankle size (units)	Mean	Std	Mean	Std
0.05,0.05	1.5	1,1,1,1	654.45	489.51	1562.55	55.35
0.1,0.1	1	0.5,1,1,0.5	285.43	459.85	1205.53	165.51
0.1,0.4	1.5	1,1,1,1	568.51	462.51	1644.79	151.35

Table 1: Comparison of pre-trained Ant agent with the MAML Ant agent for several configurations in the rllab Ant environment after meta-testing, averaged across 50 episodes



(a) Low friction (b) Asymmetry in ankle sizes (c) Asymmetry in friction and increased torso size

Figure 2: Average rewards comparison of the pre-trained policy against MAML policy during meta-testing across gradient updates, for different environment configurations

The videos containing our experimental results that can be found **here** (URL: https://bit.ly/2jvW3mM) should help the reader infer further about the exact difference in behavior of the agent in the discussed configurations. For the low friction setting, the pre-trained agent is unable to adapt its actuation to movement in the desired direction and ends up moving slower and round-about fashion also drifting slightly. The MAML agent in the same setting is able to quickly adapt its behavior to accommodate for the lower traction and stabilizes its movement resulting in less drifting. For the asymmetric ankle setting, the pre-trained agent is seen to be drifting away significantly from its desired direction and starts showing completely undesired and random behaviors on many runs. The MAML agent shows significantly better performance on this asymmetric ankle setting. Note that moving left in this setting for the ant is a much harder task due to the swollen ankles on the left. Due to such asymmetry in the agent geometry, some MAML sub-policies can perform better than the others in this case. Finally, in the asymmetric friction setting, the pre-trained agent once again drifts away from its desired direction and this behavior is worsened by the increased torso size and it is harder for the agent to stabilize its movement. Similar to all the previous cases, the MAML agent adapts to this task quite well and performs as expected.

We compared the performance of master policy in solving the Locomotion + Maze hierarchical task when using pre-trained sub-policies against using MAML trained sub-policies. The results are shown in Table 2. Note that both the MAML sub-policies and the pre-trained sub-policies have been fine-tuned to adapt to the task settings. Master policy using MAML-trained sub-policies achieve better performance compared to when using pre-trained sub-policies. We considered three different settings where: (1) low friction setting where the friction coefficient is $(0.05, 0.05)$ with torso and ankles set at their default sizes, (2) asymmetric friction setting where the friction coefficient is $(.1, .2)$ with torso and ankles set at their default sizes, (3) the difficult setting with the friction coefficients as in (2), the four ankle sizes at 0.5, 1.5, 1.5 and 0.5 units and the torso size set at 1.5 units.

	Fraction of successful episodes		Number of steps	
	MAML	Pre-trained	MAML	Pre-trained
Low friction	.81	.65	1412 ±562	1931 ±892
Asymmetric friction	.73	.59	2049 ±847	3215 ±1460
Difficult setting	.61	.32	2611 ±1567	5201 ±2872

Table 2: Comparison of pre-trained Ant agent and MAML trained Ant agent in Locomotion + Maze hierarchical task

The performance is evaluated across 3000 episodes each lasting for a total of 10000 time steps. The number of steps in the table corresponds to the number of time steps consumed in successful episodes. In all the three settings, master policy using MAML achieves higher success rate than its pre-trained counterpart. The difference is further increased in the difficult setting. Pre-trained sub-policies are only able to achieve half of the success rate of the MAML-trained sub-policies. Furthermore, the average time steps of successful episodes is less for MAML-trained sub-policies compared to pre-trained sub-policies. This reflects the ability of MAML-trained sub-policies to adapt to the current task setting resulting in better performance of the master policy. On the other hand, pre-trained sub-policies are not able to adjust properly to the settings and hence achieve lower performance.

We have also uploaded the videos of the agent trying to solve the hierarchical task in different settings. The videos demonstrate the ability of meta-learned sub-policies in quickly adapting to the current task. On the other hand, the pre-trained sub-policies lead to significant drifting of the agent resulting in longer time steps for solving the task and also lower success rate. One of the key observations is that the agent often gets stuck at corners because of its physical structure. This is because the set of skills learned by the agent are simple movements along straight lines. With more enhanced skills like moving in an arc, the agent will be able to evade such corners and its performance is expected to improve. This area is left for future work.

7 Conclusions and Future Work

In this work, we proposed a framework for solving sparse rewards or long horizon tasks in dynamic environments. Our formulation brings together the domains of meta reinforcement learning and hierarchical reinforcement learning. We proposed a meta-learning approach for fast adaptation towards new environment configuration entailing variations in both the environment parameters such as friction and agent physique such as torso and ankle sizes. We have demonstrated the effectiveness of our approach through: (1) fast adaptation traits of our meta-learning approach in several environment configurations, (2) significant improvement both in terms of quantitative rewards and qualitative agent behavior on the sub-policy tasks compared to the poorly performing pre-trained model and (3) notably higher success rate and lower number of time-steps on the sparse-reward task compared to the pre-trained model.

As future work, we plan to investigate the effectiveness of our approach on richer dynamic environments where the environment distribution varies further. We also plan to attest the computationally expensive gradient through a gradient procedure in our meta-learning approach using a first-order approximation and analyze the difference in performance for our problem between the two approaches.

Other limitation of our current approach is having a fixed horizon length for each sub-policy. This limits the applicability of the proposed method in applying to different environments as some domain knowledge is required to determine the appropriate horizon length. This issue can be alleviated with the help of a termination policy learned by the Manager similar to the option framework. Finally, the policies are represented as feed forward neural networks hence the decision of selecting next skill is based on the sensor observation at the time of switching and not on the entire horizon. Using a Recurrent Neural Network can eliminate this limitation.

References

[1] Marcin Andrychowicz et al. “Learning to learn by gradient descent by gradient descent”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 3981–3989.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. “The Option-Critic Architecture.” In: *AAAI*. 2017, pp. 1726–1734.

[3] Marc Bellemare et al. “Unifying count-based exploration and intrinsic motivation”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 1471–1479.

[4] Samy Bengio et al. “On the optimization of a synaptic learning rule”. In: *Preprints Conf. Optimality in Artificial and Biological Neural Networks*. Univ. of Texas. 1992, pp. 6–8.

[5] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. “Intrinsically motivated reinforcement learning”. In: *Advances in neural information processing systems*. 2005, pp. 1281–1288.

[6] Antoine Cully et al. “Robots that can adapt like animals”. In: *Nature* 521.7553 (2015), p. 503.

[7] Christian Daniel, Gerhard Neumann, and Jan Peters. “Autonomous reinforcement learning with hierarchical REPS”. In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE. 2013, pp. 1–8.

[8] Thomas G Dietterich. “Hierarchical reinforcement learning with the MAXQ value function decomposition”. In: *Journal of Artificial Intelligence Research* 13 (2000), pp. 227–303.

[9] Yan Duan et al. “Benchmarking deep reinforcement learning for continuous control”. In: *International Conference on Machine Learning*. 2016, pp. 1329–1338.

[10] Yan Duan et al. “RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning”. In: *arXiv preprint arXiv:1611.02779* (2016).

[11] Benjamin Eysenbach et al. “Diversity is All You Need: Learning Skills without a Reward Function”. In: *arXiv preprint arXiv:1802.06070* (2018).

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *arXiv preprint arXiv:1703.03400* (2017).

[13] Carlos Florensa, Yan Duan, and Pieter Abbeel. “Stochastic neural networks for hierarchical reinforcement learning”. In: *arXiv preprint arXiv:1704.03012* (2017).

[14] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *arXiv preprint arXiv:1801.01290* (2018).

[15] Nicolas Heess et al. “Learning and transfer of modulated locomotor controllers”. In: *arXiv preprint arXiv:1610.05182* (2016).

[16] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. “Learning to learn using gradient descent”. In: *International Conference on Artificial Neural Networks*. Springer. 2001, pp. 87–94.

[17] Rein Houthooft et al. “Evolved Policy Gradients”. In: *arXiv preprint arXiv:1802.04821v2* (2018).

[18] Rein Houthooft et al. “Vime: Variational information maximizing exploration”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 1109–1117.

[19] Sergey Levine et al. “End-to-end training of deep visuomotor policies”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[20] Ke Li and Jitendra Malik. “Learning to optimize”. In: *arXiv preprint arXiv:1606.01885* (2016).

[21] Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[22] Nikhil Mishra et al. “Meta-learning with temporal convolutions”. In: *arXiv preprint arXiv:1707.03141* (2017).

[23] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), p. 529.

[24] Ronald Parr and Stuart J Russell. “Reinforcement learning with hierarchies of machines”. In: *Advances in neural information processing systems*. 1998, pp. 1043–1049.

[25] Sachin Ravi and Hugo Larochelle. “Optimization as a model for few-shot learning”. In: (2016).

[26] Adam Santoro et al. “Meta-learning with memory-augmented neural networks”. In: *International conference on machine learning*. 2016, pp. 1842–1850.

[27] Stefan Schaal et al. “Learning movement primitives”. In: *Robotics research. the eleventh international symposium*. Springer. 2005, pp. 561–572.

[28] Jürgen Schmidhuber. “Learning to control fast-weight memories: An alternative to dynamic recurrent networks”. In: *Neural Computation* 4.1 (1992), pp. 131–139.

[29] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[30] John Schulman et al. “Trust region policy optimization”. In: *International Conference on Machine Learning*. 2015, pp. 1889–1897.

[31] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[32] Richard S Sutton, Doina Precup, and Satinder Singh. “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning”. In: *Artificial intelligence* 112.1-2 (1999), pp. 181–211.

[33] Emanuel Todorov, Tom Erez, and Yuval Tassa. “Mujoco: A physics engine for model-based control”. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 5026–5033.

[34] Christopher M Vigorito and Andrew G Barto. “Intrinsically motivated hierarchical skill learning in structured environments”. In: *IEEE Transactions on Autonomous Mental Development* 2.2 (2010), pp. 132–143.

[35] Jane X Wang et al. “Learning to reinforcement learn”. In: *arXiv preprint arXiv:1611.05763* (2016).