Hierarchical and Interpretable Skill Acquisition in Multi-task Reinforcement Learning

Tianmin Shu[*]
University of California, Los Angeles
tianmin.shu@ucla.edu

Caiming Xiong [†]
Salesforce Research
cxiong@salesforce.com

Richard Socher
Salseforce Research
rsocher@salesforce.com

Abstract

Learning policies for complex tasks that require multiple skills is a major challenge in reinforcement learning (RL). This paper proposes a novel framework for efficient multi-task reinforcement learning by training agents to employ hierarchical policies that decide when to use a previously learned policy and when to learn a new skill. This enables agents to continually acquire new skills during different stages of training. Each learned task corresponds to a human language description, so the agent can always provide a human-interpretable description of its choices. In order to help the agent learn the complex temporal dependencies necessary for the hierarchical policy, we provide it with a stochastic temporal grammar. We validate our approach on Minecraft games explicitly designed to test the ability to reuse previously learned skills while simultaneously learning new skills.

1 Introduction

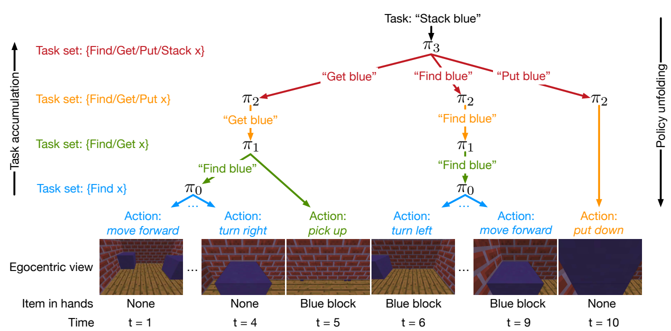
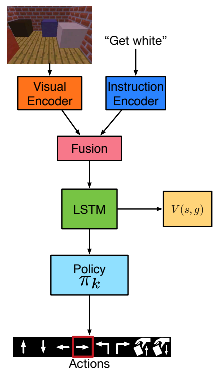


Figure 1: Example of our hierarchical policy for a given task – stacking two blue blocks. Each arrow represents one step generated by a certain policy and the colors of arrows indicate the source policies.

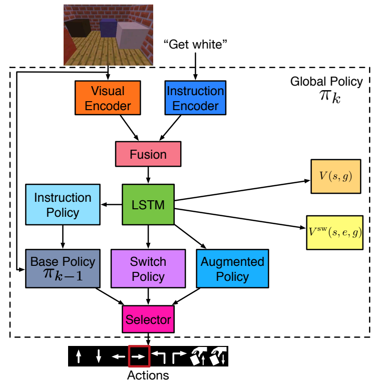
Deep reinforcement learning has demonstrated success in policy search for tasks in domains like game playing (Mnih et al., 2015; Silver et al., 2016, 2017; Kempka et al., 2016; Mirowski et al., 2017) and robotic control (Levine et al., 2016a,b; Pinto & Gupta, 2016). However, it is very difficult to accumulate multiple skills using just one policy network Teh et al. (2017). Existing approaches (Bengio, 2012; Rusu et al., 2016; Parisotto et al., 2016; Teh et al., 2017; Andreas et al., 2017) usually treat all tasks independently. This often prevents full exploration of the underlying relations between different tasks.

[*]This work was done when the author was an intern at Salesforce Research.
[†]Corresponding author.

Hierarchical Reinforcement Learning Workshop at the 31st Conference on Neural Information Processing Systems (HRL@NIPS 2017), Long Beach, CA, USA.



(a) Flat policy.



(b) Hierarchical design of global policy.

Figure 2: Flat and hierarchical policy architectures.

In this work, we propose a hierarchical policy network which can reuse previously learned skills alongside and as subcomponents of new skills. To represent the skills and their relations in an interpretable way, we also encode all tasks using human instructions. This allows the agent to communicate its policy and generate plans using human language. Figure 1 illustrates an example: given the instruction "Stack blue," our hierarchical policy learns to compose instructions and take multiple actions through a multi-level hierarchy in order to stack two blue blocks together. In order to better track temporal relationships between tasks, we train a stochastic temporal grammar (STG) model on the sequence of policy selections (previously learned skill or new skill) for positive episodes.

We validated our approach by testing it on object manipulation tasks implemented in a Minecraft world. The results show that this framework can (i) efficiently learn hierarchical policies and representations for multi-task RL; and ii) learn to utter human instructions to deploy pretrained policies, improve their explainability and reuse skills.

2 Model

2.1 Multitask RL Setting

Let $\mathcal{G}$ be a task set, where each task g is uniquely described by a human instruction. For simplicity, we assume a two-word tuple template consisting of a skill and an item for such a phrase, i.e., $g = \langle u_{\text{skill}}, u_{\text{item}} \rangle$. For each task, we define a Markov decision process (MDP) represented by states $s \in \mathcal{S}$, primitive actions $a \in \mathcal{A}$, and a reward function $R(s, g)$. As a starting point, we have a terminal policy π_0 (as shown in Figure 2a) trained for a set of basic tasks (i.e., a terminal task set $\mathcal{G}_0$). The task set is then progressively increased at multiple stages, such that $\mathcal{G}_0 \subset \mathcal{G}_1 \subset \cdots \subset \mathcal{G}_K$, which results in life-long learning of polices from π_0 for $\mathcal{G}_0$ to π_K for $\mathcal{G}_K$.

2.2 Hierarchical Policy

Instead of using a flat policy (Figure 2a), we propose a hierarchical design (Figure 2b) with the ability to reuse the base policy (i.e., π_{k-1}) for performing base tasks as subtasks. This hierarchy consists of four sub-policies: a base policy, an instruction policy, an augmented flat policy, and a switch policy.

The base policy is defined to be the global policy at the previous stage $k-1$. The instruction policy informs base policy π_{k-1} which base tasks it needs to execute. We define this policy using two conditionally independent distributions, i.e., $\pi_k^{\text{inst}}(g' = \langle u_{\text{skill}}, u_{\text{item}} \rangle | s, g) = p_k^{\text{skill}}(u_{\text{skill}}|s, g) p_k^{\text{item}}(u_{\text{item}}|s, g)$. An augmented flat policy, $\pi_k^{\text{aug}}(a|s, g)$, ensures that the global policy is able to perform novel tasks in $\mathcal{G}_k$ that can not be achieved by only reusing the base policy. A switch policy, $\pi_k^{\text{sw}}(e|s, g)$, determines whether to perform a base task or perform a primitive action, where e is a binary variable indicating the selection of the branches, π_k^{inst} $(e = 0)$ or π_k^{aug} $(e = 1)$.

At each time step, we first sample e_t from our switch policy π_k^{sw} and a new instruction g'_t from our instruction policy π_k^{inst}. Based on e_t and g'_t, the action is then sampled by

$$a_t \sim \pi_k(a_t|s_t, g) = \pi_{k-1}(a_t|s_t, g'_t)^{(1-e_t)} \pi_k^{\text{aug}}(a_t|s_t, g)^{e_t}, \tag{1}$$

where π_k and π_{k-1} are the global policies at stage k and $k-1$ respectively. After each step, we will also obtain a reward $r_t = R(s_t, g)$.

2.3 Stochastic Temporal Grammar

Inspired by previous research (Si et al., 2011; Pirsiavash & Ramanan, 2014) on stochastic grammar models, we summarize temporal transitions between various tasks with an stochastic temporal grammar (STG). In our full model, the STG interacts with the hierarchical policy described above through modified switch policy and instruction policy by using the STG as a prior.

In an episode, the temporal sequence of e_t and g'_t, $\{\langle e_t, g'_t \rangle; t \geq 0\}$, can be seen as a finite state Markov chain (Baum & Petrie, 1966). At each level $k > 0$, we may define an STG of a task g by i) transition probabilities, $\rho_k(e_t, g'_t | e_{t-1}, g'_{t-1}, g)$, and ii) the distribution of $\langle e_0, g'_0 \rangle$, $q_k(e_0, g'_0 | g)$.

With the estimated probabilities, we sample e_t and g'_t in an episode at level $k > 0$ w.r.t. to reshaped policies $\pi_k^{sw'}$ and $\pi_k^{\text{inst}'}$ respectively: if $t = 0$,

$$e_0 \sim \pi_k^{\text{sw}'}(e_0 | s_t, g) \propto \pi_k^{\text{sw}}(e_0 | s_t, g) \textstyle\sum_{g' \in \mathcal{G}_{k-1}} q_k(e_0, g' | g), \tag{2}$$

$$g'_0 \sim \pi_k^{\text{inst}'}(g'_0 | s_t, g) \propto \pi_k^{\text{inst}}(g'_0 | s_t, g) q_k(e_0 = 0, g'_0 | g); \tag{3}$$

otherwise,

$$e_t \sim \pi_k^{\text{sw}'}(e_t | e_{t-1}, g'_{t-1}, s_t, g) \propto \pi_k^{\text{sw}}(e_t | s_t, g) \textstyle\sum_{g' \in \mathcal{G}_{k-1}} \rho_k(e_t, g' | e_{t-1}, g'_{t-1}, g), \tag{4}$$

$$g'_t \sim \pi_k^{\text{inst}'}(g'_t | e_{t-1}, g'_{t-1}, s_t, g) \propto \pi_k^{\text{inst}}(g'_t | s_t, g) \rho_k(e_t = 0, g'_t | e_{t-1}, g'_{t-1}, g). \tag{5}$$

2.4 Plan Composition

Combined with our hierarchical policy and STG, we are able to run an episode to compose a plan for a task specified by a human instruction. Note that to fully utilize the base policy, we assume that once triggered, a base policy will play to the end before the global policy considers the next move.

2.5 Learning

We learn our final hierarchical policy through k stages of skill acquisition. A 2-phase curriculum learning is applied to each of these stages. In phase 1, we only sample tasks from the base task set $\mathcal{G}_{k-1}$. In phase 2, we sample tasks from the full task set, $\mathcal{G}_k$, for the k-th stage of skill acquisition.

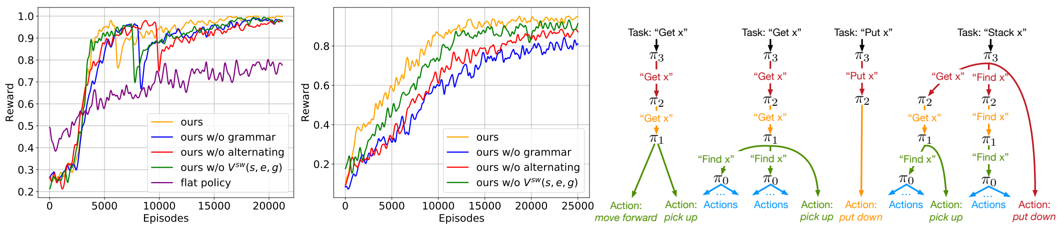
We use advantage actor-critic (A2C) for policy optimization with off-policy learning (Su et al., 2017). Let $V_k(s_t, g)$ be a value function indicating the expected return given state s_t and task g. To reflect the nature of the branch switching in our model, we introduce another value function $V_k^{\text{sw}}(s_t, e_t, g)$ to represent the expected return given state s_t, task g and current branch selection e_t. Thus, given a trajectory $\Gamma = \{\langle s_t, e_t, g'_t, a_t, r_t, \mu_k^{\text{sw}}(\cdot|s_t), \mu_k^{\text{inst}}(\cdot|s_t, g), \mu_k^{\text{aug}}(\cdot|s_t, g), g \rangle : t = 0, 1, \cdots, T\}$ generated by old policies $\mu_k^{\text{sw}}(\cdot|s_t)$, $\mu_k^{\text{inst}}(\cdot|s_t, g)$, and $\mu_k^{\text{aug}}(\cdot|s_t, g)$, the policy gradient reweighted by importance sampling can be formulated as

$$\begin{aligned}
&\underbrace{\omega_t^{\text{sw}} \nabla_{\theta^{\text{sw}}} \log \pi_k^{\text{sw}}(e_t | s_t, g) A(s_t, g, e_t)}_{\text{1st term: switch policy gradient}} + \underbrace{(1 - e_t) \omega_t^{\text{inst}} \nabla_{\theta^{\text{inst}}} \log \pi_k^{\text{inst}}(g'_t | s_t, g) A(s_t, g, e_t, g'_t)}_{\text{2nd term: instruction policy gradient}} \\
+ \; &\underbrace{e_t \omega_t^{\text{aug}} \nabla_{\theta^{\text{aug}}} \log \pi_k^{\text{aug}}(a_t | s_t, g) A(s_t, g, e_t, a_t)}_{\text{3rd term: augmented policy gradient}},
\end{aligned} \tag{6}$$

where $\omega_t^{\text{sw}} = \frac{\pi_k^{\text{sw}}(e_t|s_t,g)}{\mu_k^{\text{sw}}(e_t|s_t,g)}$, $\omega_t^{\text{inst}} = \frac{\pi_k^{\text{inst}}(g'_t|s_t,g)}{\mu_k^{\text{inst}}(g'_t|s_t,g)}$, and $\omega_t^{\text{aug}} = \frac{\pi_k^{\text{aug}}(a_t|s_t,g)}{\mu_k^{\text{aug}}(a_t|s_t,g)}$ are importance sampling weights for the three terms respectively; $A(s_t, g, e_t)$, $A(s_t, g, e_t, g'_t)$, and $A(s_t, g, e_t, a_t)$ are estimates of advantage functions, which have multiple possible definitions. In this paper, we define them as: $A(s_t, g, e_t) = \sum_{\tau=0}^{\infty} \gamma^\tau R(s_{t+\tau}, g) - V_k(s_t, g)$, $A(s_t, g, e_t, g'_t) = A(s_t, g, e_t, a_t) = \sum_{\tau=0}^{\infty} \gamma^\tau R(s_{t+\tau}, g) - V_k^{\text{sw}}(s_t, g, e_t)$, where γ is the discounted coefficient.

Finally, the value functions can be updated using the following gradient:

$$\nabla_{\theta_v} \frac{1}{2} \left[\sum_{\tau=0}^{\infty} \gamma^\tau R(s_{t+\tau}, g) - V_k(s_t, g) \right]^2 + \nabla_{\theta_v^{\text{sw}}} \frac{1}{2} \left[\sum_{\tau=0}^{\infty} \gamma^\tau R(s_{t+\tau}, g) - V_k^{\text{sw}}(s_t, e_t, g) \right]^2 / 2. \tag{7}$$



(a) Learning curves for π_1. (b) Learning phase 2 for π_3. (c) Examples of composed hierarchical plans.

Figure 3: Results: i) comparison of learning efficiency on two task sets: $\mathcal{G}_1$ for global policy π_1 (a) and $\mathcal{G}_3$ for global policy π_3 (b) respectively, and ii) typical composed hierarchical plans (c).

ρ_k and q_k of the STG are both initialized to be uniform distributions. When the agent receives a positive reward after an episode, we use maximum likelihood estimation (MLE) to update the distributions of the STG (Baum & Petrie, 1966). As the training progresses, the STG starts to guide the exploration. We use ϵ-greedy to avoid falling into local minima in the early stages of training.

3 Experiments

Game environment and task specifications. We created an indoor environment in Minecraft using the Malmo platform (Johnson et al., 2016). In each episode, an arbitrary number of blocks with different colors (totally 6 colors in our experiments) are randomly placed in a room. We consider four sets of tasks: i) $\mathcal{G}^{(0)} = \{\text{“Find x”}\}$, walking to the front of a block with color x, ii) $\mathcal{G}^{(1)} = \{\text{“Get x”}\}$, picking up a block with color x, iii) $\mathcal{G}^{(2)} = \{\text{“Put x”}\}$, putting down a block with color x, and iv) $\mathcal{G}^{(3)} = \{\text{“Stack x”}\}$, stacking two blocks with color x together. When reaching the goal of a task, the agent gets a $+1$ reward. We assume the following skill acquisition order: $\mathcal{G}_k = \cup_{\kappa=1}^{k} \mathcal{G}^{(\kappa)}$, $\forall k = 0, 1, 2, 3$, which is a natural way to increase skill sets. This results in policies $\{\pi_k : k = 0, 1, 2, 3\}$ for these four task sets.

Implementation details. The visual and instruction encoding (bag-of-words) modules have the same architectures as the ones in Hermann et al. (2017). The fusion layer simply concatenates the outputs of these two modules and feeds its output to an LSTM with 256 hidden units. We train the network with RMSProp (Tieleman & Hinto, 2012) with a learning rate of 0.0001. We set the batch size to be 36 and clip the gradient to a unit norm. For all tasks, the discounted coefficient is $\gamma = 0.95$. For the 2-phase curriculum learning, the reward threshold is 0.9. We apply ϵ-greedy to the decision sampling for the global policy, where ϵ gradually decreases from 0.1 to 0.

Results. To evaluate the learning efficiency, we compare our full model with 1) a flat policy (Figure 2a) as in Hermann et al. (2017) fine-tuned on π_0 and variants of our approach: ours without 2) STG, 3) alternating updates, or 4) $V_k^{\text{sw}}(s, e, g)$. In Figure 3a, we use various methods to train policy π_1 for the task set $\mathcal{G}_1$ based on the same base policy π_0. The large dip in the reward indicates that the curriculum learning switches from phase 1 to phase 2. To further examine the learning efficiency during phase 2, we first pretrain π_3 following our definition of phase 1 in the curriculum learning. We then proceed to learning phase 2 using different approaches all based on this pretrained policy (Figure 3b). As shown in Figure 3a and Figure 3b, our full model has the fastest convergence. Figure 3c visualizes a few typical hierarchical plans. To evaluate the generalization of learned policies, we also train the flat policy and our full model for $\mathcal{G}_1$ with only 1 item in the room and test them in a room with multiple items. Both of them achieve near perfect success rate in training scenarios. In testing cases, the flat policy has only 29% success rate, whereas our full model maintains a 94% success rate. This shows that the hierarchical policy generalizes better as it inherits the concept of items from its base policy.

4 Conclusion

In this work, we have proposed a hierarchal policy modulated by an STG as a novel framework for efficient multi-task reinforcement learning via multiple training stages. Experiments on Minecraft games have shown that our full model i) has a significantly higher learning efficiency than a flat policy does, ii) generalizes well in unseen environments, and iii) is able to compose hierarchical plans in an interpretable manner.

References

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning (ICML)*, 2017.

Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *JMLR: Workshop on Unsupervised and Transfer Learning*, 2012.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojtek Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Tocze, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games (CIG)*, 2016.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016a.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, , and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. In *" International Symposium on Experimental Robotics (ISER)*, 2016b.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. In *International Conference on Learning Representations (ICLR)*, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *IEEE Conference on Robotics and Automation (ICRA)*, 2016.

Hamed Pirsiavash and Deva Ramanan. Parsing videos of actions with segmental grammars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *International Conference on Learning Representations (ICLR)*, 2016.

Zhangzhang Si, Mingtao Pei, Benjamin Yao, and Song-Chun Zhu. Unsupervised learning of event and-or grammar and semantics from video. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, and Julian Schrittwieser et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Van Den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *The 18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*, 2017.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Tijmen Tieleman and Geoffrey Hinto. Lecture 6.5—rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.