Master-Slave Curriculum Design for Reinforcement Learning

Yuechen Wu, Wei Zhang*, Ke Song
School of Control Science and Engineering, Shandong University
{wuyuechen, songke_vsislab}@mail.sdu.edu.cn, davidzhangsdu@gmail.com

Abstract

Curriculum learning is often introduced as a leverage to improve the agent training for complex tasks, where the goal is to generate a sequence of easier subtasks for an agent to train on, such that final performance or learning speed is improved. However, conventional curriculum is mainly designed for one agent with fixed action space and sequential simple-to-hard training manner. Instead, we present a novel curriculum learning strategy by introducing the concept of master-slave agents and enabling flexible action setting for agent training. Multiple agents, referred as master agent for the target task and slave agents for the subtasks, are trained concurrently within different action spaces by sharing a perception network with an asynchronous strategy. Extensive evaluation on the ViZDoom platform demonstrates the joint learning of master agent and slave agents mutually benefit each other. Significant improvement is obtained over A3C in terms of learning speed and performance.

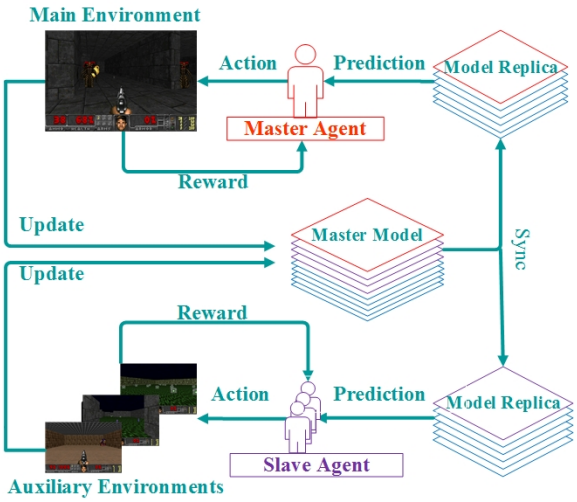

Figure 1: The proposed curriculum learning architecture. The master and slave agents run with different frequencies in the asynchronous process, where the master agent takes half of the threads totally, while all slave agents share the rest half threads.

1 Introduction

Recently, deep learning has achieved considerable success in various fields of intelligent tasks such as navigation [Jaderberg *et al.*, 2016] and texture classification [Zhang *et al.*, 2018b]. With aid of deep learning, the Deep Q-Network (DQN) [Mnih *et al.*, 2015] and the asynchronous advantage actor-critic (A3C) [Mnih *et al.*, 2016] outperformed expert human performance on Atari 2600 games. Levine *et al.* presented an end-to-end scheme to learn a visuomotor control policy for robotic grasping [Levine *et al.*, 2016]. For the foreseeable future, deep reinforcement learning will to be one of most promising parts of artificial intelligence systems.

The recent trends of reinforcement learning focus on dealing with increasingly complicated tasks. However, learning control policy directly from the complex environments with sparse feedback is very challenging. One pioneering method was presented in [Wu and Tian, 2017], which integrated reinforcement learning with curriculum learning [Jiang *et al.*, 2015] for the complex video game Doom, and finally won the champion of Track1 in Visual Doom AI Competition 2016 [Kempka *et al.*, 2016]. Andreas *et al.* used the curriculum learning scheme [Bengio *et al.*, 2009] to make their model scale up smoothly from simple tasks to difficult ones with short sketches [Andreas *et al.*, 2016]. In summary, the core idea of curriculum learning is to split particularly complex task into smaller, easier-to-solve ones [Florensa *et al.*, 2017; Kumar *et al.*, 2010]. The target agent is enabled to reject examples which it currently considers too hard, and becomes more skilled gradually as the difficulty of the tasks increases. However, the above reinforcement learning methods share the following limitations. First, the subtasks in a curriculum are all designed to train one agent (*i.e.*, the target agent), where the tasks mainly change in difficulty level and the goal is the same, *e.g.*, using a simpler map for battle in Doom. Hence, the action space is fixed among different tasks, and the agent should be trained sequentially, *i.e.*, one task after another. Second, when training with a simpler task of the curriculum, the samples generated from the corresponding easier environment normally have strong correlation. Thus, the algorithm

*Corresponding author.

is easily trapped into local minima, which affects the convergence speed and performance seriously. All these issues pose great challenges to the application of curriculum learning to reinforcement learning.

To relieve these limitations, we intend to design a flexible curriculum by introducing the concept of master-slave agents with multiple action spaces as illustrated in Figure 1. Specifically, the target task is first decomposed into a group of subtasks to complete in different environments. Unlike conventional curriculum, two types of agents are introduced in this work: the master agent for the target task and the slave agents for the subtasks. In general, the target task is harder than the subtasks, and thus each slave agent is derived from the master agent, so does the action space. Hence, the multiple agents here may have different actions to learn. The slave agents trained on the subtasks are leveraged to improve the training of the master agent on the target task.

To transfer knowledge among different tasks more effectively, all agents are trained concurrently with a shared perception network [Zhang *et al.*, 2018a] in an asynchronous manner, which differs much from the conventional curriculum learning that trains an agent one task after another. The network parameters are updated simultaneously by the asynchronous agents of different tasks. Similar to conventional curriculum learning, multiple auxiliary environments are needed for subtask training. Nevertheless, the auxiliary environments here are agent-specific with different goals, and thus the environment difference is mainly about the task goals and more like inter-class difference. Hence, coupled with the asynchronous training strategy, better exploration and exploitation could be attained by transferring the knowledge among different environments.

Evaluation is conducted on the *VizDoom* platform which offers a variety of scenarios for agent training. The inputs of the tested model are only the raw frame observations without any other game variables revealing the internal state of the game. A3C is employed as the baseline algorithm to build the whole reinforcement learning framework. Experimental results on Doom show that with the proposed curriculum learning architecture, the performance of A3C could be improved significantly in both convergence rate and number of monsters killed.

2 Related Work

In the past years, curriculum learning has been introduced into the reinforcement learning to enable the agent to learn an optimal policy in complex environments. Most practical curriculum-based approaches in reinforcement learning relied on pre-specified task sequences with manually designed schedule [Karpathy and Van De Panne, 2012]. Recently, Narvekar $et\ al.$ [Narvekar *et al.*, 2017] and Svetlik $et\ al.$ [Svetlik *et al.*, 2017] attempted to generate the task sequence of a curriculum automatically, and found that inappropriate or wrongly ordered curriculum schedules may hurt the agent training. To summarize, the current curriculum learning studies mainly focused on either the content/order of the schedule or the manner to generate schedule, such as manually [Narvekar *et al.*, 2016] or automatically [Florensa *et al.*, 2017]. They share the similarity that the schedule was designed for one agent with fixed action space. In contrast, we intend to extend the curriculum learning by introducing a master-slave agent concept which can allow flexible action setting and training manner.

This differs from the conventional multi-agent work [Busoniu *et al.*, 2007], which aimed to train an ensemble of self-interested and independent agents to learn their own policies by maximizing a discounted sum of rewards. Thus, each agent has to consider its teammates' behavior and to find a cooperative policy. Instead, the multiple agents here have the master-slave configuration. As illustrated in Figure 3, each slave agent is derived from the master agent. It aims to boost the learning of the master agent, though has its own task. The master-slave agents care trained jointly by sharing a perception network with an asynchronous manner. Also, the proposed architecture is compatible to most reinforcement learning schemes and can work together at low computational cost.

The essence and purpose of our method also differ from the conventional multi-task learning [Wilson *et al.*, 2007; Li *et al.*, 2009]. For example, multiple DDPG networks were employed in [Yang *et al.*, 2017] to learn multiple tasks that are equally important, such as going forward and backward. In [Jaderberg *et al.*, 2016], multiple auxiliary tasks and related pseudo-rewards were exploited to facilitate the target task training without any extra environment. In contrast, herein a series of different subtasks and task-specific environments are introduced (for slave agents) to optimize performance for a specific target task (for master agent), rather than all tasks. The slave agent interacting with agent-specific environment was apt to receive extrinsic rewards.

Besides, the role of the subtasks in our method is different from the hierarchical reinforcement learning. Such as in [Tessler *et al.*, 2017], the subtask was pre-trained as a skill for the target agent and the skill will be reused when a similar subtask exists. In [Kulkarni *et al.*, 2016], subtasks were constructed priori for the target task and a top-level value function learns a policy over them. But in our method, subtasks and target task were trained concurrently for different agents, which could mutually benefit each other. Also, our subtasks are unnecessary when the training is finished.

3 Proposed Curriculum Learning

In this section, we present a reinforcement learning architecture based on A3C, and extend the curriculum learning to allow multiple agents with flexible action setting. As illustrated in Figure 1, the agents are defined with the configuration of master and slave to learn their control policies within different action spaces. The knowledge among them is transfered by a shared perception network with an asynchronous manner. The ultimate goal is to improve the learning of the master agent and make it perform better in the target task.

3.1 Preliminaries

Reinforcement learning [Sutton and Barto, 1998] is concerned with training an agent interacting with an environment to maximize the expected future rewards. The environments are defined as a Markov Decision Process (MDPs), which is

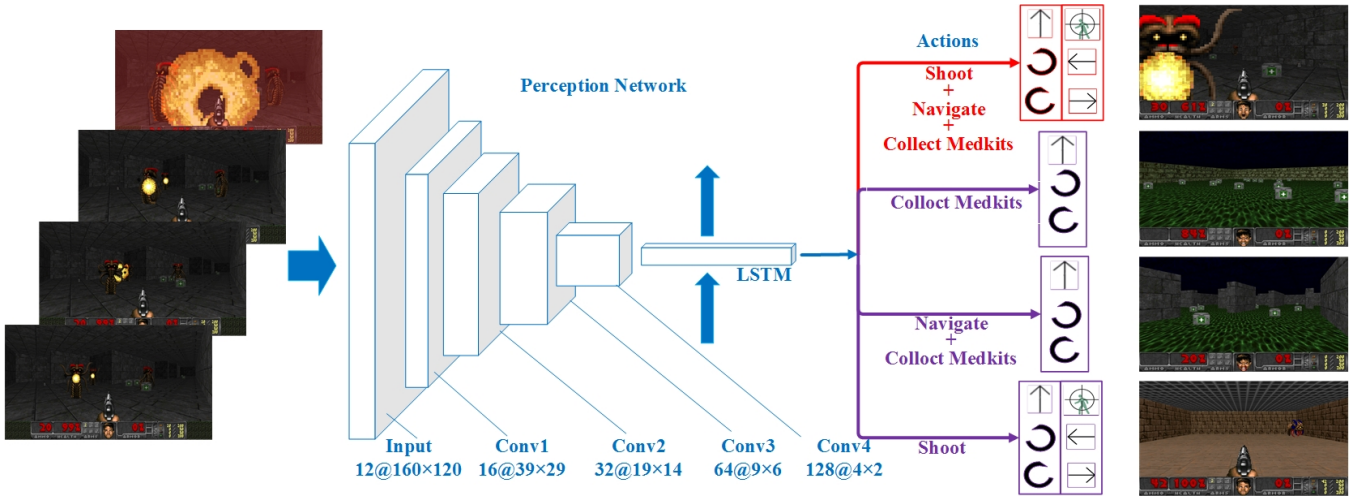

Figure 2: Illustration of the proposed deep reinforcement learning model for master-slave agent training. The model consists of two parts: a shared perception network built by CNNs and LSTM, and k decision networks with different output sizes for different actions. Note that only one actor will be activated to choose actions in each timestep.

formalized as a tuple(s, a, r, γ), where s denotes a finite state space, a denotes the action space, r denotes the state-reward function and $\gamma \in (0, 1]$ is a discount factor. A deterministic policy $\pi(a|s)$ maps each state to an action and defines the behavior of the agent.

At each discrete time step t, the agent observes the state s_t and chooses an action a_t according to its policy $\pi(a_t|s_t)$. One time step later, the agent receives a numerical reward r_t and finds itself in a new state s_{t+1}. The process continues until the agent reaches a terminal state. The return R_t is the total accumulated rewards from time step t. The goal of the agent is to learn an optimal control policy π, which maximizes its expected return until the episode ends.

A3C is an actor-critic algorithm which updates both the policy function $\pi(a_t|s_t;\theta_\pi)$ and the state-value function $V(s_t;\theta_v)$ by n-step returns. The policy and the value function are updated after every t_{max} actions or when a terminal state is reached. The total accumulated return from time step t is defined as:

$$R_t = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{k+t};\theta_v) \tag{1}$$

where k can vary from state to state and is upper-bounded by t_{max}.

The entropy H of the policy π is added to the objective function to alleviate premature convergence to suboptimal deterministic policies. The gradient of the full objective function can be regarded as:

$$\begin{aligned}\nabla_{\theta_\pi} log\pi(a_t|s_t;\theta_\pi)(R_t - V(s_t;\theta_v)) \\ +\beta\nabla_{\theta_\pi} H(\pi(s_t;\theta_\pi)),\end{aligned} \tag{2}$$

where β controls the strength of the entropy regularization term.

The final gradient update rules are listed as follows:

$$\begin{aligned}\theta_\pi \leftarrow \theta_\pi + \eta\nabla_{\theta_\pi} log\pi(a_t|s_t;\theta_\pi)(R_t - V(s_t;\theta_v)) \\ +\beta\nabla_{\theta_\pi} H(\pi(s_t;\theta_\pi)).\end{aligned} \tag{3}$$

$$\theta_v \leftarrow \theta_v + \eta\nabla_{\theta_v}(R_t - V(s_t;\theta_v))^2, \tag{4}$$

where η represents the learning rate.

3.2 Curriculum Design

When training an agent to perform a complex task, conventional curriculum learning methods first generate a few easier tasks ($e.g.$, using an easier map or reducing the number of adversaries), and then train the agent one by one with increasing difficulty levels, while the actions to learn are the same.

Given a target task, we also decompose it into multiple subtasks, however, the subtasks are coupled with additional agents which are referred as the slave ones. Accordingly, the original agent is regarded as the master one. The curriculum here is designed for multiple agents to learn the corresponding tasks in their own environments. Therefore, the action spaces differ from each other as well. Taking Doom for example shown in Figure 3, the goal of the master agent is to kill the monsters as much as possible with less health loss. Thus, the target task can be decomposed into three subtasks including shooting, navigation, and collecting medkits. For the subtask of shooting (bottom right), only monsters exist and the corresponding slave agent is trained to learn how to kill them only. Hence, the subtasks of a curriculum are designed for the slave agents respectively (rather than the master one), that have different actions to learn. To share the knowledge among different subtasks, the slave agents are trained jointly with the master agent in a deep model explained in the following sections.

3.3 Architecture Design

As shown in Figure 2, we present a deep model based on the actor-critic architecture. The input is a stack of sequential images ($e.g.$, four frames) captured from the environments, which are partially observable. The model consists of two parts: a shared perception network and k decision networks with different output sizes for different action spaces. The

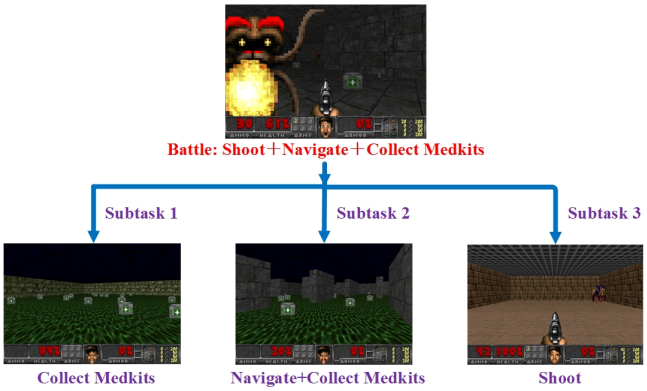

Figure 3: Task decomposition in Doom. Top is the "*Battle*" where the agent defends against adversaries while gathering health and ammunition in a maze; Bottom left is "*Collect Medkits*" where the agent gathers health kits in a square room (without navigation); Bottom middle is "*Collect Medkits and Navigate*" where the agent gathers health kits in a maze (with navigation); Bottom right is the "*Shoot*" where the agent aims at shooting monsters.

perception network is composed of a four-layer convolutional neural network (CNN) and a long short-term memory (LSTM) network by which the raw environment observations can be transformed into state representations. The decision networks have two types of outputs: policy and value. The policy decides the action probability distribution for an agent to choose, and the value is a score about the state. In the training and testing, the environment and task obey the rule of one-to-one correspondence. Given the observation about an environment, only the action of corresponding task can be predicted.

3.4 Asynchronous Learning Strategy

The proposed asynchronous learning strategy is illustrated in Figure 1. Rather than training the agent sequentially in a curriculum, the master-slave agents are trained to learn the corresponding control policies concurrently with an asynchronous manner. It is noted that the master and slave agents run with different frequencies in the asynchronous process, where the master agent takes half of the threads and the slave agents share the rest half threads.

When one agent interacts with its corresponding environment, the agents for the other tasks may have not finished the network updating yet. In a single update, the actions are selected based on the predicted policy for up to t_{max} steps or until a terminal state is reached. Such procedure makes the agent receive up to t_{max} rewards from the environment since last update. Then gradients are computed for n-step update for each of the state encountered. The longest possible n-step return is employed, $i.e.$, a one-step update for the last state, a two-step update for the second last state, and so on for a total of up to t_{max} updates. Then, the accumulated updates are applied in a single gradient step which is referred as a minibatch.

Therefore, the shared perception network could learn the state representation jointly across all data of the tasks, including target task and subtasks. On the other hand, such asynchronous training manner may make the perception network update more frequently than the decision networks. To address this, different learning rate setting is employed in the experiments. The initial learning rate of the perception network η^p is sampled from a $LogUniform(10^{-4}, 10^{-2})$ distribution and annealed to 10^{-4} with the global shared counter T over the course of training. The initial learning rates of the decision networks η_i^d are also sampled from this distribution but annealed with the independent task shared counter T_i according to the corresponding tasks, where k represents the number of the tasks and $i \in [1, k]$. In the experiments, RMSProp was performed to optimize the network in TensorFlow. Without locking, the moving average of elementwise squared gradients g is shared among threads and updated asynchronously. The standard non-centered RMSProp update with momentum α and exploration rate ϵ is given by:

$$g_{net} = \alpha g_{net} + (1 - \alpha)\Delta\theta_{net}^2 \tag{5}$$

$$\theta_{net} \leftarrow \theta_{net} - \eta_{net}\frac{\Delta\theta_{net}}{\sqrt{g_{net}+\epsilon}} \tag{6}$$

where net represents the perception network or decision networks. Algorithm 1 gives an summary of the proposed algorithm.

Algorithm 1: Master-Slave Curriculum Learning

//Assume global shared perception network parameter vectors θ^p and global shared counter $T = 0$
//Assume global shared decision network parameter vectors $\theta_1^d, \cdots, \theta_i^d$ and global shared task counter $T_1, \cdots, T_i = 0$, where $i \in [1, k]$
//Assume thread-specific weights $\theta^{p'}$ and $\theta_i^{d'}$
Initialize thread step counter $t \leftarrow 1$
while $T_i \leqslant$ Threshold$_i$ **do**
  Reset gradients: $d\theta^p \leftarrow 0$ and $d\theta_i^d \leftarrow 0$.
  Synchronize weights $\theta^{p'} = \theta^p$ and $\theta_i^{d'} = \theta_i^d$
  $t_{start} = t$
  Get state s_t
  while terminal s_t **or** $t - t_{start} == t_{max}$ **do**
    Perform a_t according to policy $\pi(a_t|s_t; \theta^{p'}, \theta_i^{d'})$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T$
    $T_i \leftarrow T_i + 1$
  end
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta^{p'}, \theta_i^{d'}) & \text{non-terminal } s_t \end{cases}$
  for $i \in \{t-1, \cdots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt $d\theta^p$ and $d\theta_i^d$ using equ. 3
  end
  Perform asynchronous updates of θ^p using $d\theta^p, \eta^p$, and of θ_i^d using $d\theta_i^d, \eta_i^d$.
end

4 Experimental Results

In this section, we test the proposed architecture to explore the following questions with the aid of *VizDoom* platform that uses the first-person perspective in a complex and dynamic 3D environment.

- Does the proposed curriculum improve the performance of reinforcement learning?
- Is the master-slave agent configuration effective in curriculum learning?
- What is the effect of asynchronous joint learning on master-slave agent training?

4.1 Evaluation and Setting

As illustrated in Figure 3, four scenarios of *VizDoom* were employed to train agents in the experiments as follows.

1. *Collect Medkits:* An agent is spawned randomly in a square room and loses its health slowly and constantly. To survive as long as possible, the agent needs to move around and collect medkits, which appear in random places during the episode. Each episode ends after 10500 ticks (1 second = 35 ticks) or when the agent dies. There are three actions to learn: move forward, turn left, and turn right.
2. *Collect Medkits and Navigate:* Except the agent is spawned randomly in a maze, the others are the same to those of *Collect Medkits*, including the actions to learn.
3. *Shoot:* An agent is spawned at one side of a square room, and a stationary monster is spawned randomly along the other side. A single hit is enough to kill the monster. Each episode ends after 2100 ticks or when the monster dies. There are six actions to learn: move forward, turn left, turn right, move left, move right, and shoot.
4. *Battle:* An agent and the monsters are spawned randomly in a maze. The monsters move around in the maze, and shoot fireballs at the agent. The agent defends against monsters, and collects medkits and ammunitions, which appear randomly during the episode. Each episode ends after 10500 ticks or when the agent dies. The actions to learn are the same to those of *Shoot*.

For all experiments, we set the discount factor $\gamma = 0.99$, the RMSProp decay factor $\alpha = 0.99$, the exploration rate $\epsilon = 0.1$, and the entropy regularization term $\beta = 0.01$. To reduce the computational burden, the agent received an input every m game frames, where m denotes the number of frames skipped. In the experiment, we used 16 threads and performed updates after every 80 actions ($i.e.$, $t_{max} = 20$ and $m = 4$).

4.2 Effect of Curriculum on Master Agent

In this section, experiments were conducted to test and analyze the proposed curriculum learning architecture against the baseline A3C method. As shown in Figure 4, we first train the master agent individually based on A3C for the target task. Then the master agent and the slave agents are also jointly trained with different combinations based on the proposed curriculum learning architecture.

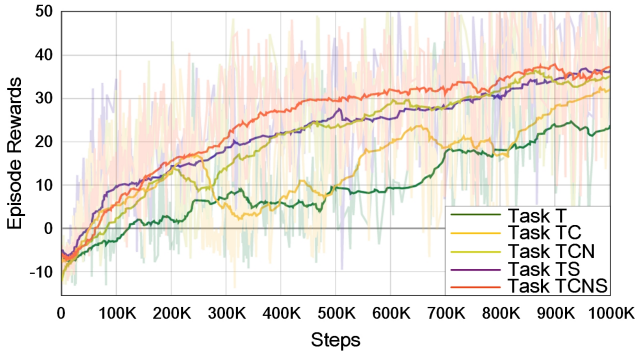

Figure 4: Effect of curriculum to master agent by setting "*Battle*" as the target task which is labeled as T. The subtasks include "*Collect Medkits*", "*Collect Medkits and Navigate*" and "*Shoot*" which are labeled as C, CN and S.

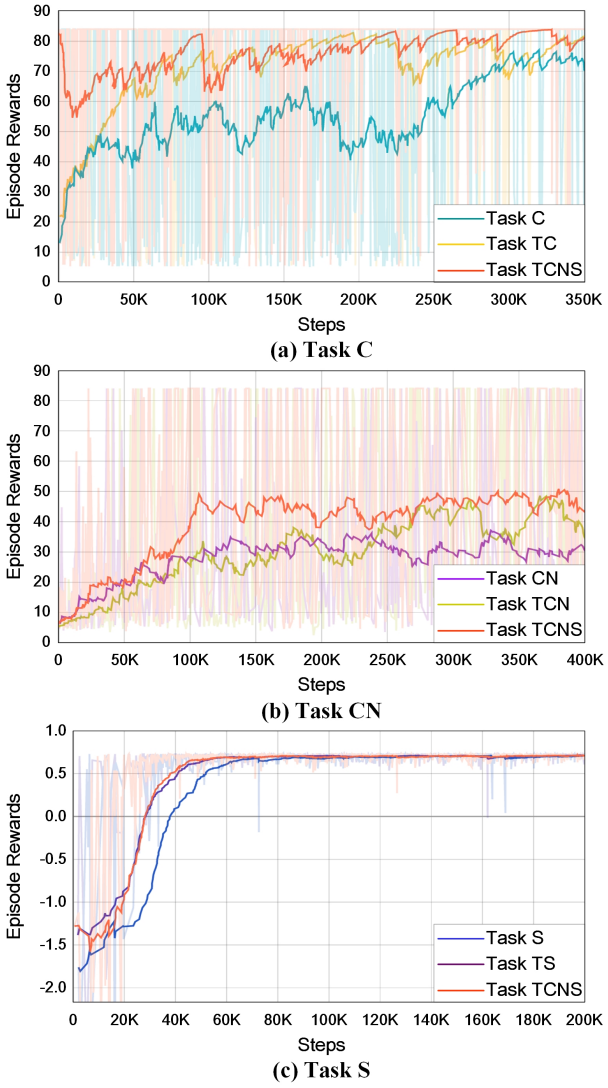

Figure 5: Effect of curriculum to slave agents by setting "*Battle*" as the target task and different subtasks as: "*Collect Medkits*" in (a), "*Collect Medkits and Navigate*" in (b), and "*Shoot*" in (c).

Specifically, for the target task "*Battle*", the master agent trained together with the slave agents did perform better than the one trained individually with A3C, *i.e.*, faster learning speed and higher episode rewards. The best master agent was obtained with all subtasks involved. The results demonstrate that the slave agents could boost the master agent via the proposed asynchronous curriculum. Also, it can be concluded that the proposed reinforcement learning architecture is more flexible and remains stable to the curriculum changes in subtask as well as action space. It is worth noting that the increased number of subtasks do not increase the number of episodes needed to stabilize training. This may be owing to the perception network shared among all tasks, which may help the agent avoid undesired actions and increase the chance of collecting rewards during exploration.

4.3 Effect of Curriculum on Slave Agent

Figure 5(a)-(c) show the effects of the proposed curriculum on the three subtasks. Similarly, we observed that training a slave agent with the master agent and the other slave agents converged faster and obtained higher rewards than training it individually with its subtask only by A3C. This could be referred as a byproduct of our curriculum learning algorithm.

It may be due to the following reasons: First, the subtask is simpler and the samples for training are highly correlated, which may make reinforcement learning easily trapped into local minimal and affect the convergence speed; Second, additional samples could be provided from different environments to boost training; Finally, the inter-class difference among environments may encourage more exploration. For example, the subtask "*Collect Medkits*" in (a) is much simpler than the target task "*Battle*", as the agent only needs to learn three moving actions in a square room. After including "*Battle*" as the target task for joint training, the performance of "*Collect Medkits*" was improved significantly.

4.4 Discuss on Master-Slave Configuration

Additional experiments were conducted to further validate the master-slave configuration as illustrated in Figure 6. First, since the subtask "*Collect Medkits*" is derived partially from the "*Collect Medkits and Navigate*", where the agent also needs to navigate in a maze besides collecting medkits. Hence, we can set the "*Collect Medkits and Navigate*" as the target task to train a master agent, and "*Collect Medkits*" as a subtask to train a slave agent. Also, the subtask "*Collect Medkits*" is replaced by "*Shoot*" to discuss the difference.

It can be concluded that training "*Collect Medkits and Navigate*" with "*Collect Medkits*" performed better than individual training with A3C. Since the two environments are similar and not much additional rewards can be yielded, the improvement is not that evident as most above results. Besides, we also observed that the joint training with "*Shoot*" did not produce any gains and the algorithm even cannot reach convergence. This is because these two tasks are completely different and can be regarded as parallel ones, which cannot satisfy the our definition on master and slave. What we stress here is that the the slave agent should originate from the master agent, so does the tasks. Hence, appropriate master-slave configuration is desired for a good curriculum.

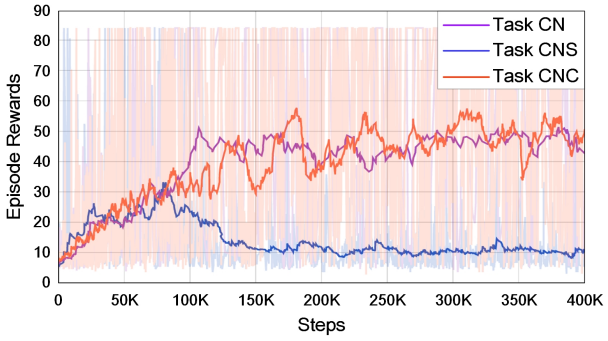

Figure 6: Analysis on master-slave configuration by setting "*Collect Medkits and Navigate*" as the target task (originally subtask 2). The original Subtask 1 "*Collect Medkits*" and Subtask 3 "*Shoot*" are included to train the slave agents, respectively.

	Frags	Items
Target Task	52.2	9.6
Target Task+Subtask 1	64.2	27.1
Target Task+Subtask 2	65.1	21.2
Target Task+Subtask 3	79.4	18.5
Target Task+Subtask 1-3	82.9	31.6

Table 1: Performance evaluation on "*Battle*" in Item and Frag.

4.5 Performance Evaluation on *Battle*

The performance of agent training on "*Battle*" could be evaluated with two measures: Item (number of medkits and ammunitions picked) and Frag (number of monsters killed). Table 1 reports the performance of the proposed architecture for training an agent in "*Battle*", where each fully trained agent was tested for five thousands episodes. The average Item and Frag were reported in the table. Apparently, benefiting from curriculum learning with a slave agent, the capability of the master agent for a specific task was enhanced. For example, assisted by the shooting-learner with subtask 3, the master agent becomes better at killing monsters and yielded higher Frag counts than training itself individually. Meanwhile, it was also improved at collecting medkits and ammunitions, when jointly training with subtask 1 and subtask 2. Similarly, the best agent was obtained when all subtasks were learned concurrently with the proposed asynchronous curriculum.

5 Conclusions

In this paper, we presented a new curriculum learning architecture and introduced it to A3C to train agents in complex tasks. The proposed curriculum was produced by decomposing the target task into related subtasks coupled with multiple auxiliary agents referred as slave agents with flexible action setting. To transfer the knowledge among different tasks and boost the master agent, a shared perception network was proposed, and optimized concurrently by all agents with an asynchronous manner. Results on Doom demonstrated the effectiveness of the proposed master-slave curriculum strategy.

Acknowledgements

This work was supported by the NSFC Grant No.61573222, Shenzhen Future Industry Special Fund JCYJ20160331174228600, Major Research Program of Shandong Province 2015ZDXX0801A02, National Key Research and Development Plan of China under Grant 2017YFB1300205 and Fundamental Research Funds of Shandong University 2016JC014.

References

[Andreas *et al.*, 2016] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. *arXiv preprint arXiv:1611.01796*, 2016.

[Bengio *et al.*, 2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48, 2009.

[Busoniu *et al.*, 2007] Lucian Busoniu, Robert Babuska, and Bart De Schutter. Multi-agent reinforcement learning: A survey. In *International Conference on Control, Automation, Robotics and Vision*, pages 1–6, 2007.

[Florensa *et al.*, 2017] Carlos Florensa, David Held, Markus Wulfmeier, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017.

[Jaderberg *et al.*, 2016] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[Jiang *et al.*, 2015] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *AAAI*, page 6, 2015.

[Karpathy and Van De Panne, 2012] Andrej Karpathy and Michiel Van De Panne. Curriculum learning for motor skills. In *Canadian Conference on Artificial Intelligence*, pages 325–330, 2012.

[Kempka *et al.*, 2016] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8, 2016.

[Kulkarni *et al.*, 2016] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NIPS*, pages 3675–3683, 2016.

[Kumar *et al.*, 2010] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, pages 1189–1197, 2010.

[Levine *et al.*, 2016] 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. *The International Journal of Robotics Research*, 2016.

[Li *et al.*, 2009] Hui Li, Xuejun Liao, and Lawrence Carin. Multi-task reinforcement learning in partially observable stochastic environments. *Journal of Machine Learning Research*, 2009.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Mnih *et al.*, 2016] 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 *ICML*, pages 1928–1937, 2016.

[Narvekar *et al.*, 2016] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. Source task creation for curriculum learning. In *AAMAS*, pages 566–574, 2016.

[Narvekar *et al.*, 2017] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *IJCAI*, pages 2536–2542, 2017.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[Svetlik *et al.*, 2017] Maxwell Svetlik, Matteo Leonetti, Jivko Sinapov, Rishi Shah, Nick Walker, and Peter Stone. Automatic curriculum graph generation for reinforcement learning agents. In *AAAI*, pages 2590–2596, 2017.

[Tessler *et al.*, 2017] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*, volume 3, page 6, 2017.

[Wilson *et al.*, 2007] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *ICML*, pages 1015–1022, 2007.

[Wu and Tian, 2017] Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. *ICLR*, 2017.

[Yang *et al.*, 2017] Zhaoyang Yang, Kathryn Merrick, Hussein Abbass, and Lianwen Jin. Multi-task deep reinforcement learning for continuous action control. In *IJCAI*, pages 3301–3307, 2017.

[Zhang *et al.*, 2018a] Wei Zhang, Qi Chen, Weidong Zhang, and Xuanyu He. Long-range terrain perception using convolutional neural networks. *Neurocomputing*, 275:781–787, 2018.

[Zhang *et al.*, 2018b] Wei Zhang, Weidong Zhang, Kan Liu, and Jason Gu. A feature descriptor based on local normalized difference for real-world texture classification. *IEEE Transactions on Multimedia*, 20(4):880–888, 2018.