Auxiliary selection: optimal selection of auxiliary tasks using deep reinforcement learning

Hidenori Itaya† Tsubasa Hirakawa† Takayoshi Yamashita† Hironobu Fujiyoshi†
† Chubu University

Abstract

A method using auxiliary task is a type of multi-task learning. This improves the performance of the target task by simultaneously learning auxiliary task. However, this method requires that the auxiliary task must be effective for the target task. It is very difficult to determine in advance whether a designed auxiliary task is effective, and the effective auxiliary task changes dynamically according to the learning status of the target task. Therefore, we propose an auxiliary task selection mechanism, Auxiliary Selection, based on deep reinforcement learning. We confirmed the effectiveness of our method by introducing it to UNREAL, a method that has achieved high agent performance by introducing auxiliary tasks.

Keywords: Deep learning, Auxiliary task, Reinforcement learning

1. Introduction

Real-world problems are complex mixtures of various elements, and even seemingly completely different tasks can be closely related. Solving these different tasks simultaneously with a single model can improve model performance and reduce training and inference time. This learning method is called multi-task learning, and it can efficiently learn features that are common to multiple different tasks. For example, in computer vision, solving object detection and semantic segmentation simultaneously has been reported to improve the performance of both tasks [1, 2].

Auxiliary learning is a type of multi-task learning. This is a learning method that improves the performance of the main task as a kind of normalization by adding auxiliary tasks that are not related to the main task to be solved to the target task and learning the auxiliary tasks at the same time. In automated driving, it has been reported that the accuracy of the main task can be improved by using depth estimation and semantic segmentation from in-vehicle camera images as the main task and introducing time and weather estimation as auxiliary tasks [3]. In game playing, game score improvement has been confirmed by introducing three different auxiliary tasks, with game playing by deep reinforcement learning as the main task [4]. However, since all these auxiliary tasks are designed manually, the designed auxiliary tasks don't necessarily contribute to solving the various main tasks. On the contrary, some auxiliary tasks prevent learning depending on the main task. This problem could be solved by carefully designing auxiliary tasks that depend on the main task, but validating the effectiveness of auxiliary tasks is expensive.

In this paper, we target auxiliary learning and aim at avoiding interference with learning by unsuitable auxiliary tasks. Therefore, we propose a new auxiliary task called "Auxiliary selection," which adaptively selects auxiliary tasks according to the main task. Auxiliary selection is designed as a deep reinforcement learning agent that outputs weights for each auxiliary loss and adaptively selects auxiliary tasks for network update according to the main task. Auxiliary selection's network is trained simultaneously with other networks and shares the reward signal with the main task to find the appropriate auxiliary task according to the learning stage of the main task. We applied auxiliary selection to UNREAL, a method that introduces auxiliary tasks in game strategy, and analyzed the selected auxiliary tasks and game scores to demonstrate the selection of suitable auxiliary tasks for the main task.

The contributions of this study are as follows.

- Our method efficiently improves the performance of the main task to select the optimal auxiliary task for the main task and its training stage.
- Our method automatically suppresses unnecessary auxiliary tasks for the main task, thus reducing the design cost of auxiliary tasks.

2. Related work

In the research field of multi-task learning, there is a performance improvement of the main task by introducing auxiliary tasks. Liebel *et al.* have improved the accuracy of the main task in automated driving by using semantic segmentation and depth estimation as the main task and learning the auxiliary tasks of time and weather estimation in parallel [3]. Jaderberg *et al.* propose Unsupervised reinforcement learning and auxiliary learning (UNREAL), which learns auxiliary tasks simultaneously with the main task in deep

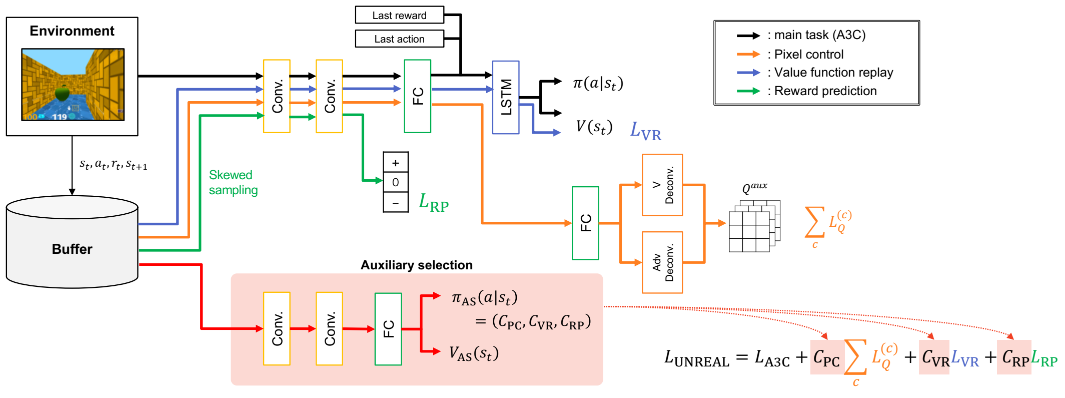

Figure 1: UNREAL with Auxiliary selection.

reinforcement learning [4]. They use three different auxiliary tasks in their work: Pixel control learns actions in which pixels in the input image change significantly, Value function replay shuffles past experiences and learns a state value function $V(s)$, Reward prediction predicts rewards to be obtained in the future. By introducing these auxiliary tasks, they have achieved high scores on the maze task of DeepMind Lab. The main task of this method is a video game strategy using Asynchronous Advantage Actor-Critic (A3C) [5].

Auxiliary learning by introducing auxiliary tasks improves performance on the main task. However, if auxiliary tasks that are not suitable for learning the main task are used, these tasks prevent the learning of main task and reduces its accuracy. Therefore, it is necessary to introduce auxiliary tasks that are suitable for the main task, and several works have been reported to solve this problem. Teh *et al.* use distillation to learn policy that are common multiple tasks, thereby avoiding interference with the main task and stabilizing learning [6]. This method achieves robustness and stability across different tasks by introducing restrictions such that the shared policy does not deviate from the effective policy for all tasks. Du *et al.* blocked training data acquired by an auxiliary task that was not effective [7]. This method focuses on the gradient of losses in the main and auxiliary tasks and uses the auxiliary task losses if the cosine similarity between the gradients is high, and blocks the auxiliary task training data if it is low. Riedmiller *et al.* propose Scheduled auxiliary control (SAC-X) [8], which selects a policy to solve the main task from multiple low-level policies. Their method designs several low-level auxiliary tasks, and each auxiliary task learns a policy that is aligned with a low-level target. Lin *et al.* calculate the gradient similarity between the main and auxiliary losses using a Taylor approximation involving the gradient of the auxiliary loss [9]. The method determines the auxiliary task to be used when learning the main task based on the calculated gradient similarity. On the other hand, we consider the purpose of the auxiliary task to be to improve the accuracy of the main task, and design a deep reinforcement learning agent that controls the auxiliary loss with the same reward signal as the main task, i.e., to improve the accuracy of the main task, independent of other tasks. This agent is trained in parallel with the main task to achieve dynamic selection of the auxiliary task.

3. UNREAL with Auxiliary selection

The effectiveness of the auxiliary task depends on the main task, and an unsuitable auxiliary task may prevent learning of the main task. Therefore, we propose an Auxiliary selection that adaptively selects an auxiliary task according to the main task. In this section, we describe our method in detail by applying it to UNREAL.

3.1. Auxiliary selection

Figure 1 shows the network structure of our method. Our method was built based on UNREAL, and we added the auxiliary selection. In the auxiliary selection, images stored in a replay buffer are input and then a state value $V_{\mathrm{AS}}(s)$ and a policy π_{AS} are output. Among them, the policy π_{AS} represents whether we use each auxiliary task for the main task training or not. Here, we denote weights for each task as $C_{\mathrm{PC}} = \{0, 1\}$, $C_{\mathrm{VR}} = \{0, 1\}$, and $C_{\mathrm{RP}} = \{0, 1\}$. The suffixes are PC for Pixel control, VR for Value function replay, and RP for Reward prediction. The policy π_{AS} is defined as

$$\pi_{\mathrm{AS}} = (C_{\mathrm{PC}}, C_{\mathrm{VR}}, C_{\mathrm{RP}}). \tag{1}$$

Unlike other auxiliary tasks, the network for auxiliary selection is not shared with the network of the main task, it train independently. Thus, we adaptively select auxiliary tasks depending on the environment.

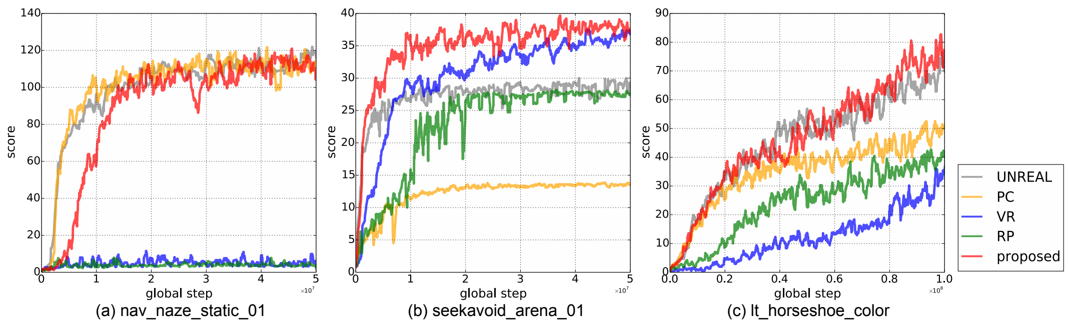

Figure 2: The game scores of DeepMind Lab over different global steps. The horizontal axis shows the number of global steps to update network parameters and the vertical axis shows scores of each task.

3.2. Loss function

We formulated the loss function of our method by using the loss function of conventional UNREAL as follows:

$$L = L_{\text{A3C}} + C_{\text{PC}} \sum_{c} L_Q^{(c)} + C_{\text{VR}} L_{\text{VR}} + C_{\text{RP}} L_{\text{RP}}, \quad (2)$$

where L_{A3C} is a loss value of the main task (i.e., A3C), and $\sum_c L_Q^{(c)}, L_{\text{VR}}$, and L_{RP} are loss values of each auxiliary task. Note that, in terms of the pixel control, we split an input image into $n \times n$ grid and compute losses for each grid. Hence, $L_Q^{(c)}$ represents the loss of n-step Q-learning for a grid c. In our method, we select auxiliary tasks by multiplying auxiliary loss and binary weight obtained from auxiliary selection.

In case that we train the auxiliary selection by using Eq. (2) simultaneously, the network is trained so that $C_{\text{VR}}, C_{\text{PC}}$, and C_{RP} become zero. Therefore, we define another loss function to train the network of the auxiliary selection and train the network apart from training the main and auxiliary tasks. At the same time, the rewards used for training the auxiliary selection are the same as those used for the main task. In this way, the auxiliary selection controls the weights of the auxiliary tasks to improve the accuracy of the main task. The loss function of the auxiliary selection can be formulated using loss functions of the state value $V_{\text{AS}}(s)$ and the policy $\pi_{\text{AS}}(a|s)$ as follows:

$$L_{\text{AS}v} = (r + \gamma V_{\text{AS}}(s_{t+1}, \theta^-) - V_{\text{AS}}(s_t, \theta))^2 \quad (3)$$

$$L_{\text{AS}p} = -\log(\pi_{\text{AS}}(a|s))A(s,a) - \beta H(\pi_{\text{AS}}), \quad (4)$$

where θ^- is the network parameters before a network update, r is the same rewards of main task, $A(s,a)$ is advantage function. And, an entropy $H(\pi_{\text{AS}})$ promotes explorations that prevent the network parameters from converging into a local minima, and β is a scale parameter for the entropy $H(\pi_{\text{AS}})$.

Finally, the loss function of auxiliary selection is defined by adding losses of Eqs. (3) and (4) as

$$L_{\text{AS}} = L_{\text{AS}v} + L_{\text{AS}p}. \quad (5)$$

4. Experiments

We used the DeepMind Lab [10] for an evaluation of our method. DeepMind Lab mainly contains three games: i) nav_maze_static_01 (maze), ii) seekavoid_arena_01 (seekavoid), and iii) lt_horseshoe_color (horseshoe).

We compared our method with the following baselines: UNREAL (Three auxiliary tasks are used for training), PC, VR, and RP (Each of them uses an auxiliary task for training, respectively). The common hyperparameters during training were unified. The training steps are 5.0×10^7 steps for maze and seekavoid, and 1.0×10^8 steps for horseshoe.

4.1. Comparison by game scores

maze. Figure 2(a) shows scores for the maze. UNREAL and PC achieved higher performances, while scores of VR and RP are almost zero. This means that VR and RP did not improve the main task. The PC promoted an agent to take action changing pixel values. In other words, this enabled an agent to move in every corner of the maze environment. Our method also achieved a higher score as UNREAL and PC.

seekavoid. Figure 2(b) shows the score of the seekavoid. PC was inadequate for the seekavoid because pixel values changed significantly even when negative rewards were obtained. RP was also not efficient because this environment is dense rewards. On the other hand, UNREAL and VR achieved higher scores. Surprisingly, VR outperformed UNREAL, and our method also achieved higher performance as VR.

horseshoe. Figure 2(c) shows the score of the horseshoe. PC was the most high score in horseshoe. The reason is that actions that defeat enemies change pixel values significantly. However, UNREAL outperforms the other methods, and our method achieved the same performance as UNREAL.

Table 1: The number of times and percentage each auxiliary task was selected in one episode.

Env.	Auxiliary task		
	PC	VR	RP
maze	435.4 (48.3%)	487.8 (54.1%)	369.0 (41.0%)
seekvoid	0.3 (0.1%)	300.0 (100.0%)	0.0 (0.0%)
horseshoe	8545.1 (94.9%)	14.1 (0.1%)	8998.2 (99.9%)

4.2. Analysis of the selected auxiliary tasks

Table 1 shows the number of times and the percentage each auxiliary task was selected in one episode. Note that the number of selected auxiliary tasks was calculated by averaging over 50 episodes. The number of action steps in an episode was 900 for the maze, 300 for seekavoid, and 9,000 for horseshoe.

The results of the maze show that all auxiliary tasks were equivalently selected. Because appropriate auxiliary tasks for the maze task were UNREAL or PC, our method equally selected all auxiliary tasks. In seekavoid, our method stably selected the value function replay. Since these results correspond to the results in Fig. 2(b), our method only selects auxiliary tasks that contribute to the training of the main task. In horseshoe, pixel control and reward prediction were often selected. Although the best score was achieved by UNREAL, auxiliary selection for horseshoe did not select value function replay. To analyze the reason of the selection, we conducted additional experiments. In addition to the results of baselines shown in Fig. 2(c), we added the following baselines: A3C (Without auxiliary tasks), PC+RP (Uses pixel control and reward prediction). Figure 3 shows the scores of each baseline and our method. This results shows that the score of VR was lower than that of A3C. And PC+RP achieved the same score as UNREAL and our method. Therefore, our method successfully removes the value function replay from the training of horseshoe.

These results above show that our approach can select auxiliary tasks that contribute to training the main task.

5. Conclusion

In this paper, we propose an auxiliary selection that dynamically selects auxiliary tasks according to the main task. Auxiliary selection is a deep reinforcement learning agent that controls auxiliary losses in order to improve the performance of the main task. It dynamically selects the auxiliary task according to the learning situation of the main task. Experimental results show that our method can train the network to select appropriate auxiliary tasks and solve the main task efficiently. As future work, we plan to apply our method to other auxiliary learning methods and to experiment with various tasks.

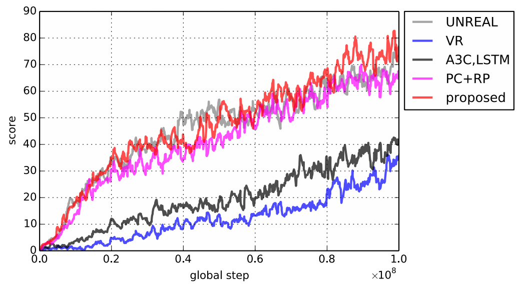

Figure 3: Game score in horseshoe with additional auxiliary task combinations.

References

[1] M. Teichmann, M. Weber, et al., “Multinet: Real-time joint semantic reasoning for autonomous driving,” In IV, pp.1013–1020, (2018).

[2] Y. Qian, J.M. Dolan, and M. Yang, “Dlt-net: Joint detection of drivable areas, lane lines, and traffic objects,” In T-ITS, vol.21, no.11, pp.4670–4679, (2020).

[3] L. Liebel and M. Körner, Auxiliary Tasks in Multi-task Learning, arXiv preprint, arXiv:1805.06334, (2018).

[4] M. Jaderberg, V. Mnih, et al., Reinforcement Learning with Unsupervised Auxiliary Tasks, In ICLR, (2017).

[5] V. Mnih, A.P. Badia, et al., “Asynchronous methods for deep reinforcement learning,” In ICML, pp.1928–1937, (2016).

[6] Y. Teh, V. Bapst, et al., “Distral: Robust multitask reinforcement learning,” In NeurIPS, pp.4496–4506, (2017).

[7] Y. Du, W.M. Czarnecki, et al., Adapting Auxiliary Losses Using Gradient Similarity, arXiv preprint, arXiv:1812.02224, (2018).

[8] M. Riedmiller, R. Hafner, et al., Learning by Playing – Solving Sparse Reward Tasks from Scratch, In ICML, (2018).

[9] X. Lin, H. Baweja, et al., “Adaptive auxiliary task weighting for reinforcement learning,” In NeurIPS, pp.4772–4783, (2019).

[10] C. Beattie, J.Z. Leibo, et al., DeepMind Lab, arXiv preprint, arXiv:1612.03801, (2016).