Applying Hybrid Reward Architecture to a Fighting Game AI

Yoshina Takano
Graduate School of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
email is0263fs@ed.ritsumei.ac.jp

Wenwen Ouyang
College of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
email is0444pk@ed.ritsumei.ac.jp

Suguru Ito
Graduate School of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
email is0202iv@ed.ritsumei.ac.jp

Tomohiro Harada
College of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
email harada@ci.ritsumei.ac.jp

Ruck Thawonmas
College of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
email ruck@ci.ritsumei.ac.jp

***Abstract*—In this paper, we propose a method for implementing a competent fighting game AI using Hybrid Reward Architecture (HRA). In 2017, an AI using HRA developed by Seijen *et al.* achieved a perfect score of 999,990 in Ms. Pac-Man. HRA decomposes a reward function into multiple components and learns a separate value function for each component in the reward function. Due to reward decomposition, an optimal value function can be learned in the domain of Ms. Pac-Man. However, the number of actions in Ms. Pac-Man is only limited to four (Up, Down, Left, and Right), and till now whether HRA is also effective in other games with a larger number of actions is unclear. In this paper, we apply HRA and verify its effectiveness in a fighting game. For performance evaluation, we use FightingICE that has 40 actions and has been used as the game platform in the Fighting Game AI Competition at CIG since 2014. Our experimental results show that the proposed HRA AI, a new sample AI for the competition, is superior to non-HRA deep learning AIs and is competitive against other entries of the 2017 competition.**

***Index Terms*—deep learning, Hybrid Reward Architecture, muti-head agents, fighting game AI, FightingICE**

I. Introduction

Reinforcement learning (RL) successfully works in many games, but sometimes it is very slow and unstable to learn the optimal value function in other some games. Atari 2600 game Ms. Pac-Man was also one of the games where the optimal value function is difficult to learn, but Seijen *et al.* [1] applied Hybrid Reward Architecture (HRA) to their proposed game AI, which won a perfect score of 999,990. This result showed the effectiveness of HRA for RL. In HRA, a reward function is decomposed into multiple components, and HRA learns a value function for each component reward function. Because each component typically only depends on a subset of all the features, the overall value function is much smoother and can be easier approximated by a low-dimensional representation, enabling more effective learning. However, the number of actions in Ms. Pac-Man is only four (Up, Down, Left, and Right), and there has been no work showing whether HRA is also effective in other games with a large number of actions. We focus on fighting games, which in general have a much higher number of actions, apply HRA to a fighting game AI and propose a method for implementing a competent game AI.

Mnih *et al.* [2] achieved a big breakthrough by their Deep Q-Network (DQN). Their AI using DQN outperformed humans on a large number of Atari 2600 games, by learning a policy directly from pixels. Justesen *et al.* [3] showed how macromanagement decisions in StarCraft can be learned directly from game replays using deep learning. However, deep learning has not yet been successfully applied to fighting games in terms of performance. Yoon and Kim [4] applied DQN to a fighting game, but their result, against a non-moving opponent AI, only showed the potential of the DQN approach in this game genre. Nguyen [5] also applied a combination of RL and a Convolutional Neural Network, but the resulting AI was not able to defeat the champion of the Fighting Game AI Competition (FTGAIC) held at CIG 2015. Therefore, we consider it is worth examining if a competent fighting game AI can be implemented using HRA.

In this work, we use FightingICE which has been used as the platform in FTGAIC[1] at CIG since 2014 [6]. There are a number of existing studies using FightingICE in research as shown in the following.

Ishihara *et al.* [7] applied Monte-Carlo Tree Search (MCTS) to FightingICE and improved the AI performance by perform-

[1] http://www.ice.ci.ritsumei.ac.jp/˜ftgaic/
https://github.com/TeamFightingICE/FightingICE

978-1-5386-4359-4/18/$31.00 ©2018 IEEE

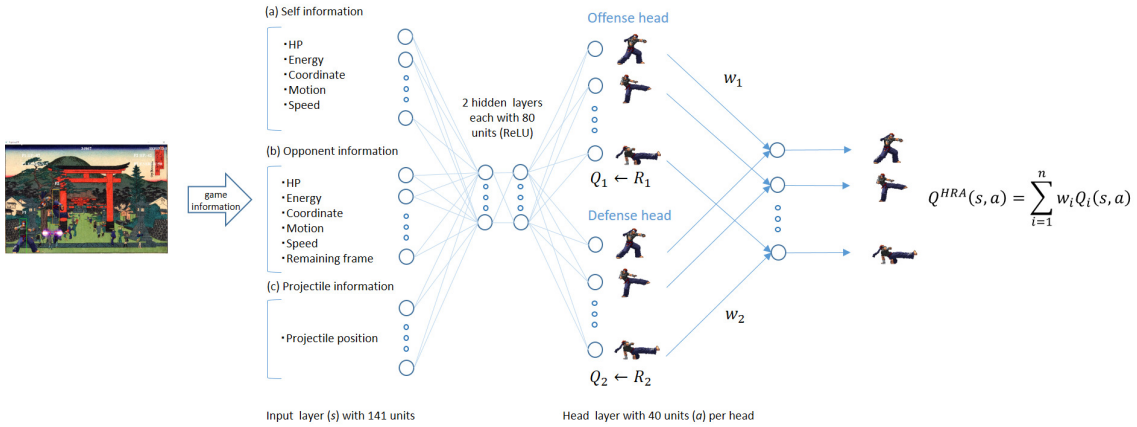

Fig. 2. HRA for FightingICE where the output layer combines the results from each of the two heads

Fig. 1. A screenshot of FightingICE

ing roulette selection in MCTS's play-out and introducing a set of rules to work with MCTS. Demediuk *et al.* [8] proposed and examined a variety of MCTS-based AIs whose aim is to adjust their difficulty level according to the player's level. Neufeld *et al.* [9] applied Hierarchical Task Network in their AI, an entry in the 2017 FTGAIC. However, their AI called HTNFighter came in 8th out of 10 entries in the competition. And more recently, Ishii *et al.* proposed a method for implementing a game AI with a persona using MCTS [10] while Ishihara *et al.* [11] not only considered difficulty adjustment but also believability in their game AI research where their AI was compared with an AI proposed by Demediuk *et al.* [8]. Figure 1 shows a screenshot of FightingICE.

II. Proposed Method

In this section, we describe a method for implementing a competent fighting game AI using HRA for FightingICE.

A. Hybrid Reward Architecture

We separate the reward function into n reward functions to learn the optimal Q-value function. The reward function of HRA is defined as follows:

$$R^{HRA}(s,a) = \sum_{i=1}^{n} w_i R_i(s,a), \quad \text{for all states } s, \text{ and actions } a, \tag{1}$$

where w_i is the weight of the each reward function. Each agent i of HRA, i.e., head i and its lower-layer structure, is trained using the respective reward function $R_i(s,a)$.

Because of individual reward function, each agent i, has its own Q-value function,. The Q-value function of HRA is defined as the weighted sum of all agents as follows:

$$Q^{HRA}(s,a;\theta) = \sum_{i=1}^{n} w_i Q_i(s,a;\theta), \tag{2}$$

where θ are training parameters. Each head can be viewed alternatively as a single DQN agent, and all the heads share multiple lower-level layers of DQN.

B. HRA for FightingICE

Figure 2 shows HRA for FightingICE. We propose using two heads: offense head and defense head. We call this multi-head AI. In FightingICE, to win the AI (self) has to give more damage to the opponent (opp) than the damage its receives. Thereby, we define the reward function as follows:

$$R_t^{HRA} = Reward_t^{offense} + Reward_t^{defense} \tag{3}$$

Here, we simply set the weight w of each head to 1. Each term on the right-hand side corresponds to the respective head and is described below where t and $t+1$ are the starting time of the current action and the subsequent action, respectively, by the AI.

1) Offense head: The offense head's role is to learn how to effectively give a damage to the opponent AI. The head obtains a reward when the AI gives a damage to the opponent AI. The reward function is defined as follows:

$$Reward_t^{offense} = HP_t^{opp} - HP_{t+1}^{opp}. \tag{4}$$

2) Defense head: The defense head's role is to learn how to effectively avoid opponent attacks. The head is penalized when the AI is hit by an opponent attack. The reward function is defined as follows:

$$Reward_t^{defense} = HP_{t+1}^{self} - HP_t^{self}. \tag{5}$$

C. Network Architecture

The network in this work is determined based on empirical results and consists of (1) an input layer with 141 units representing game information, (2) two hidden layers each with 80 units using the ReLU activation function, and (3) a head layer with two heads and 40 units per head, and (4) an output layer consisting of 40 units; in this network, the adjacent layers are fully connected. Because we can obtain the game information such as HP, Energy and coordinates of the AI and the opponent from FightingICE, we use 19 types of game-state information, leading to in total 141 input features. Note that game states can be reconstructed based on the 141 input features. FightingICE has 40 actions for controlling AIs, so each head in the head layer and the output layer have 40 units, each representing a different action. The aforementioned game information can be separated into self information, opponent information, and projectile information. We describe these features in detail below.

1) Self information: The self information consists of self HP, self Energy, self x coordinate, self y coordinate, self motion, self speed in the x direction, and self speed in the y direction; most of them are normalized to [0, 1]. However, only motion is represented as a one-hot vector of 56 units because the number of motions in FightingICE is 56.

2) Opponent information: Likewise, the opponent information consists of opponent HP, opponent Energy, opponent x coordinate, opponent y coordinate, opponent motion, opponent speed in the x direction, opponent speed in the y direction, and remaining frame (the number of remaining frames to complete the current action); accept for motion, they are normalized to [0, 1]. As done in the self information, motion is represented as a one-hot vector having 56 units.

3) Projectile information: The projectile information is composed of a projectile's x coordinate, y coordinate, and hit damage. Because there can be at most four projectiles at the same time, there are four sets of this information.

III. Experiments

In this section, we describe the details of FightingICE and the two experiments to verify the performance of the multi-head AI.

A. FightingICE

FightingICE is a real-time 2D fighting game platform. In this platform, one game consists of 60-second rounds, and one frame is set to 1/60 seconds. An AI has to decide and input its action in one frame. To make decision making by AIs challenging, they are given a game state by the system with a delay of 14 frames. Each of the two characters has two parameters: HP and Energy. Each character's initial HP is set to HP_{max} and will decrease when the respective character is hit; the initial Energy is set to 0. After 60 seconds elapse or HP of either character reaches 0, the game will proceed to the next round, and both characters' HP will be reset to HP_{max}. The character with the larger remaining HP at the end of round is the round winner. The value of HP_{max} is set to 10,000 in the first experiment to ensure the round length is fixed to 60 seconds and it is set to 400 in the second experiment according to the rule of Standard League of FTGAIC. The 2018 version of FightingICE is used in our experiments.

As described above, AIs cannot obtain game states with no delay in FightingICE. However, to construct tuples of state, action, reward, and next state used for DQN training, a mechanism is introduced, by which it is possible to determine whether an action of interest can be actually executed (in other words there are no other actions waiting to be executed) at a given – thus delayed – game state; if so, a tuple containing the action is created together with the other elements and added to a memory. We note that this mechanism is not used in any other parts in both experiments.

TABLE I
List of hyperparameters

Hyperparameter	Value
minibatch size	32
replay memory size	50000
target network update frequency	300
discount factor	0.9
learning rate	0.001
initial exploration	1
final exploration	0.1

B. Hyperparameters

We set the hyperparameters according to our empirical results. Their details are given in Table I.

C. Training against Machete

In the first experiment, we compare the proposed multi-head AI with a normal (single-head) DQN. We train both AIs for 1400 rounds against Machete that won the 2015 competition. Nguyen [5] also trained their AI against Machete, but their AI could not outperform Machete. Here, we are curious to see the performance of the multi-head AI against both single-head AI and Machete. We then analyze the learning process using the score, eqn. (6), over the number of training steps (rounds).

$$score_{self} = \frac{HP_{self}}{HP_{self} + HP_{opp}} \times a \tag{6}$$

where a is set to 1000, so the score above 500 indicates that the AI wins the opponent at the current round.

D. Comparisons with other AIs

In the second experiment, we compare the multi-head AI, the single-head AI, Machete, Ishihara's AI (ACEMctsAi) [7], and the AIs from the 2017 competition, including a sample MCTS AI (MctsAi). Here we focus on their performances in Standard League, a round-robin competition where there are two games – three rounds per game – switching sides for each pair of AIs, using a character called ZEN and its character data in the 2018 FTGAIC. Because a combo system introduced in the 2017 version has been removed, HTNFighter [9], utilizing the combo system, could not be run.

TABLE II
RESULT OF THE COMPETITION

AI name	# of wins	Win rate	Rank
GigaThunder	62	0.86	1st
MytNoAI	59	0.82	2nd
Mutagen	50	0.69	3rd
Multi-head AI	45	0.63	4th
FooAI	36	0.50	5th
Single-head AI	34.5	0.48	6th
Machete	34	0.47	7th
MegaBot	32.5	0.45	8th
MogakuMono	31	0.43	9th
ACEMctsAi	30	0.42	10th
MctsAi	26	0.36	11th
JayBot2017	19	0.26	12th
ZoneAI	9	0.13	13th

IV. Results and Discussions

In this section, we show the experimental results and our discussions in terms of whether the multi-head AI outperforms the single-head and other AIs.

A. Training against Machete

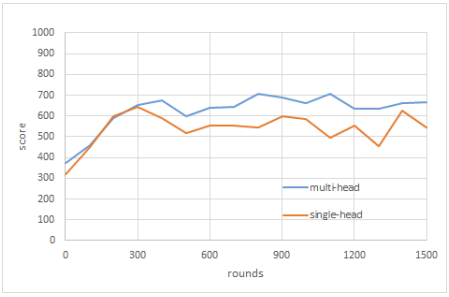

Fig. 3. Score Transitions against Machete

Figure 3 shows the average, sampled every 100 rounds, of five trials conducted in the first experiment. Both of the AIs using DQN start to defeat Machete from the 150th round, and this is the first time that DQN, to the best of our knowledge, was successfully applied in FightingICE and was able to defeat a former champion AI. The multi-head AI obtained the maximum score above 700, which outperforms the single-head. In addition, the score of the multi-head AI is more stable than that of the single-head one.

B. Comparisons with other AIs

The result of the competition is shown in Table II, where # of wins shows the number of winning rounds. The multi-head AI is the 4th among 13 AIs in Standard League. Video clips showing typical fights of the muti-head AI and the single-head AI in this competition are also available[2].

[2]http://www.ice.ci.ritsumei.ac.jp/˜ruck/HRA-cig2018.htm

V. Conclusions and Future Work

According to the experiment results, the multi-head AI obtained a higher score than the single-head AI in the training phase. In addition, the former was ranked higher than the latter in the conducted competition. The said results indicate that HRA is also effective in fighting games, where AIs conduct decision making to select their next actions from a large number of available actions. In this paper, we used only two heads. It is possible that using more heads will lead to a better performance. In addition, Machete was only used as the training opponent. The competition performance of the multi-head AI might be improved by switching the training opponents from a set of properly selected ones.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful comments. They would also like to thank their lab members, in particular, the FightingICE team members for their fruitful discussions. This research was partially supported by Strategic Research Foundation Grant-aided Project for Private Universities (S1511026), Japan.

References

[1] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, *"Hybrid reward architecture for reinforcement learning,"* in Proc. Advances in Neural Information Processing Systems. 2017. pp. 5392-5402.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, *"Human-level control through deep reinforcement learning,"* in Nature, 2015, vol. 518, pp. 529-533.

[3] N. Justesen, and S. Risi, *"Learning Macromanagement in StarCraft from Replays using Deep Learning,"* in Proc. Computational Intelligence and Games (CIG), 2017 IEEE Conference on. IEEE, 2017. pp. 162-169.

[4] S. Yoon, and K.J. Kim, *"Deep Q Networks for Visual Fighting Game AI,"* in Proc. Computational Intelligence and Games (CIG), 2017 IEEE Conference on. IEEE, 2017. pp. 306-308.

[5] D.T.T. Nguyen, *Supervised and Reinforcement Learning for Fighting Game AIs using Deep Convolutional Neural Network,* in Master Thesis, Japan Advanced Institute of Science and Technology, Mar. 2017.

[6] F. Lu, K. Yamamoto, L. H. Nomura, S. Mizuno, Y. Lee and R. Thawonmas, *"Fighting Game Artificial Intelligence Competition Platform,"* in Proc. IEEE 2nd Global Conference on Consumer Electronics (GCCE), pp.320-323, 2013.

[7] M. Ishihara, T. Miyazaki, C.Y. Chu, T. Harada, R. Thawonmas, *"Applying and Improving Monte-Carlo Tree Search in a Fighting Game AI,"* in Proc. Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology. ACM, 2016.

[8] S. Demediuk, M. Tamassia, W.L. Raffe, F. Zambetta, Xiaodong Li, *"Monte Carlo Tree Search Based Algorithms for Dynamic Difficulty Adjustment,"* in Proc. IEEE Conference on Computational Intelligence and Games (CIG 2017), New York City, USA, Aug. 22-25, 2017.

[9] X. Neufeld, S. Mostaghim, and D. Perez-Liebana, *"HTN Fighter: Planning in a Highly-Dynamic Game,"* in Proc. 2017 Computer Science and Electronic Engineering (CEEC 2017), Colchester, UK, pp. 189-194, Sep. 2017.

[10] R. Ishii, S. Ito, M. Ishihara, T. Harada, R. Thawonmas, *"Monte-Carlo Tree Search Implementation of Fighting Game AIs with Personas,"* in Proc. Computational Intelligence and Games (CIG), 2018 IEEE Conference on. IEEE, 2018.

[11] M. Ishihara, S. Ito, R. Ishii, T. Harada, R. Thawonmas, *"Monte-Carlo Tree Search for Implementation of Dynamic Difficulty Adjustment Fighting Game AIs Having Believable Behaviors,"* in Proc. Computational Intelligence and Games (CIG), 2018 IEEE Conference on. IEEE, 2018.