mathematics

MDPI

Article

Approaches That Use Domain-Specific Expertise: Behavioral-Cloning-Based Advantage Actor-Critic in Basketball Games

Taehyeok Choi, Kyungeun Cho * and Yunsick Sung *

Department of Multimedia Engineering, Dongguk University-Seoul, 30, Pildongro-1-gil, Jung-gu, Seoul 04620, Republic of Korea

* Correspondence: cke@dongguk.edu (K.C.); sung@dongguk.edu (Y.S.); Tel.: +82-2-2260-3834 (K.C.); +82-2-2260-3338 (Y.S.)

Abstract: Research on the application of artificial intelligence (AI) in games has recently gained momentum. Most commercial games still use AI based on a finite state machine (FSM) due to complexity and cost considerations. However, FSM-based AI decreases user satisfaction given that it performs the same patterns of consecutive actions in the same situations. This necessitates a new AI approach that applies domain-specific expertise to existing reinforcement learning algorithms. We propose a behavioral-cloning-based advantage actor-critic (A2C) that improves learning performance by applying a behavioral cloning algorithm to an A2C algorithm in basketball games. The state normalization, reward function, and episode classification approaches are used with the behavioral-cloning-based A2C. The results of the comparative experiments with the traditional A2C algorithms validated the proposed method. Our proposed method using existing approaches solved the difficulty of learning in basketball games.

Keywords: game AI; reinforcement learning; imitation learning; basketball game

MSC: 91A25

Citation: Choi, T.; Cho, K.; Sung, Y. Approaches That Use Domain-Specific Expertise: Behavioral-Cloning-Based Advantage Actor-Critic in Basketball Games. *Mathematics* **2023**, *11*, 1110. https://doi.org/10.3390/math11051110

Academic Editor: Sam Ganzfried

Received: 30 January 2023
Revised: 20 February 2023
Accepted: 21 February 2023
Published: 22 February 2023

1. Introduction

Recently, research on the application of artificial intelligence (AI) in games has been on the increase. Most commercial games still use AI based on a finite state machine (FSM) due to complexity and cost considerations. However, FSM-based AI decreases user satisfaction given that it performs the same patterns of consecutive actions in the same situations. Various studies have applied reinforcement learning to AI to replace a non-player character or a player [1]. Since reinforcement learning can be applied to model-free environments, it is suitable for complicated game environments. For example, a study applied a deep Q-network (DQN) [2] to Atari games [3]. The Atari game environment is effectively used to verify reinforcement learning since it provides interfaces to experiment with various games [3]. DQN succeeded in human-level AI learning by applying deep neural networks to Q-learning [4]. In addition, DQN solved the problem of complexity through a convolutional neural network (CNN).

Three heuristic pre-processes are required to successfully apply reinforcement learning in a basketball game. The first pre-process involves normalizing the state representation for learning in a complex environment of a basketball game. Since the raw data generated in the basketball game environment increases the amount of learning, it is necessary to sort selected raw data and process them to be easily understood. For example, it is necessary to reduce the state space of reinforcement learning by representation using relative values. Learning can be stabilized by normalizing raw data in an unstable basketball game environment [5]. The second pre-process is designing the reward function that

evaluates the actions of the AI. Defining the reward function to be applied to reinforcement learning in a basketball game may be difficult [6–8]. Accordingly, it is necessary to consider the correlation among various rewards generated in this complex environment during the design process. The third pre-process is classifying various episodes for learning. Learning various episodes at once increases learning complexity [7]. To classify episodes per situation, it is necessary to define and classify representative situations as offense, defense, and loose ball to reduce the complexity.

Additionally, reinforcement learning for basketball games requires an approach to improve its performance limitation given that reinforcement learning starts learning with a uniform policy and faces lots of trials and errors at the beginning of learning. Such trials and errors reduce learning performance [9]. Thus, a new AI approach that applies domain-specific expertise to existing reinforcement learning algorithms is needed.

We propose a behavioral-cloning-based advantage actor-critic (A2C), which improves learning performance by applying a behavioral cloning algorithm [10] to an A2C algorithm [11,12], a reinforcement learning algorithm, in a basketball game. We introduce three pre-processing methods, namely, state representation, reward function, and episode classification, that can be used to apply the behavioral-cloning-based A2C to a basketball game. The present study aims to solve the issues related to learning time and state space generated during reinforcement learning in complex basketball games.

This paper is configured as follows. Section 2 discusses studies related to the state representation, reward function, episode classification, and behavioral cloning algorithm. Section 3 explains the behavioral-cloning-based A2C proposed in this paper and pre-processes applied to improve reinforcement learning. Section 4 compares the performance of FSM-based AI applied to commercial games with the AI trained by the proposed method. Section 5 describes the experimental results achieved by the proposed method and areas of further study.

2. Background

Reinforcement learning is a circular structure that exchanges states, rewards, and actions, and the control is generally handled by the agent. First, the agent requests a state and reward from the environment. Furthermore, when the state and reward come from the environment, the agent determines the action and sends it back to the environment. The environment takes action, undergoes a transition to the environment, and then regenerates the state and reward in the next step.

The DQN algorithm contributed three methods to solve the problem that reward graphs fail to converge with unstable learning when combining deep neural networks and reinforcement learning. The DQN algorithm defined an image as a state in the Atari game and enabled effective learning with a relatively small image, even in the environment requiring a high-level sensor, by applying a CNN. Furthermore, the use of CNNs made the performance better as compared to the performance when the conventional simple linear function neural networks were used. The DQN algorithm used a replay buffer using previous experiences, which solved the sample correlation problem and stabilized the distribution of the data. The DQN algorithm used a target network to solve the problem of unstable target values.

The double deep Q network (DDQN) algorithm proves that the DQN algorithm can be overoptimistic. The DDQN algorithm also shows that the overestimation of the DQN algorithm is more common by analyzing the value estimation. The DDQN algorithm solved the existing problem and proved to be more stable. The DDQN algorithm proposed an implementation method that simply modifies the loss function without changing the network or other tasks. The DDQN algorithm obtained good results in the Atari 2600 domain.

The A2C algorithm is designed to reduce variance by adding the concept of a critic in the policy gradient method. There is an actor that updates parameters to find a policy and a critic that updates parameters to estimate a value. In the process of making a policy gradient, subtracting the constant value unrelated to the action can reduce the variance

without affecting the estimation. Therefore, the advantage $A(s, a)$ can be expressed as a state-action value $Q(s, a)$ and a state value $V(s)$. Additionally, applying the time difference (TD) method is efficient because the number of parameters can be reduced.

The self-imitation learning (SIL) algorithm proposed a learning method that imitates past good decisions. The SIL algorithm provided a theoretical justification for their objective function. The SIL algorithm was simply implemented and could be applied to the A2C algorithm. The SIL algorithm validated the good performance in several Atari games. The SIL algorithm was applied to the proximal policy optimization (PPO) algorithm and improved performance.

The DQN algorithm and DDQN algorithm estimate the value of actions and states to determine the next actions, which is called a value-based algorithm. It is difficult to apply the proposed method given that there is no explicit policy. The SIL algorithm imitates past decisions, which may not be enough to achieve an optimal policy. Since the A2C algorithm has an explicit policy, it is suitable to imitate an expert policy. There are limitations to the A2C algorithm itself that make it insufficient for application to complex basketball game environments, but the method presented in this paper can be improved upon.

3. Related Work

This section discusses existing reinforcement learning studies, including the state representation, reward function, and episode classification in reinforcement learning. We also introduce a behavioral cloning algorithm.

3.1. Reinforcement Learning

Reinforcement learning can be applied to various industries, such as games, stocks, and automobiles. However, in the beginning of the learning stage, reinforcement learning undergoes trials and errors, so a simulation environment is essential. Without such a simulation environment, researchers would lose huge amounts of stock in stocks or suffer from car accidents in automobiles. Since the game environment is advantageous for the application of techniques such as several times of speed and synchronization that are helpful for learning, many reinforcement learning studies have been conducted on the game environment.

Schaul et al. [13] introduced a priority-based sampling method in operating replay buffers proposed in a deep Q-network (DQN). Another study modified and improved the loss function of a DQN [14]. Another study modified and applied the network to make the state value not the state-action value in DQN [15]. There is a study that uses this distribution and performs well in certain Atari games [16]. Another study performed reinforcement learning in multiple environments using an asynchronous method [12]. In the policy-based reinforcement-learning algorithm, an algorithm applied clipping using epsilon [17]. Other studies improved an A2C algorithm using a higher TD error in the replay buffer [18]. A framework study used an Atari environment for reinforcement learning [19]. Another study defined a state and reward for applying reinforcement learning to StarCraft 2 and applied reinforcement learning to a mini-game using the reinforcement learning used in the Atari game [20]. Other studies implemented the environment for applying multi-agent reinforcement learning to StarCraft 2; they defined the state, reward, and action and experimented using various multi-agent reinforcement learning algorithms [21]. Shao et al. [6] enhanced the performance of an AI using curriculum transfer learning in the StarCraft micromanagement environment. Kurach et al. [22] developed a soccer game engine for reinforcement learning, experimented with reinforcement learning algorithms, and finally, provided an experimental environment, including a soccer game benchmark and a soccer game academy. Other studies used multi-agent reinforcement learning, position learning, episode-classification learning, and curriculum learning in basketball games [7]. Liu et al. [23] introduced a deep reinforcement learning (DRL) approach for a vehicle dispatching problem by designing the corresponding simulator and a specific

vehicle dispatching algorithm. The DRL approach simplifies the problem of the requirement for a large number of agents.

3.2. State Representation, Reward Function, and Episode Classification

State refers to environmental data given to an agent at a specific time. An agent determines the course of action based on the state. The performance of a reinforcement learning-based agent depends on the state definition [24]. Mnih et al. [2] defined an image as a state in the Atari game. When it is difficult to define an image as a state, the state is defined using values. Shao et al. [6] adopted a normalized relative coordinate value based on the agent to be learned. The merits of a normalized state are the reduction of risks of falling into local minima and making optimization by an optimizer faster [5].

Reward refers to the value assigned to the action of an agent at a specific time. Since the action policy depends on the methods defining the reward function, it is important to define the reward function [8]. Ng et al. [8] verified the change of the action policy of an agent depending on the reward. Shao et al. [6] demonstrated that the winning rate of an agent depended on reward in a StarCraft micromanagement environment. Even with the same algorithm, the learning performance converged faster with a denser reward.

Jia et al. [7] introduced a learning method that classified episodes per situation when it was difficult to learn for the entire episode and experimentally verified the performance improvement.

3.3. Behavioral-Cloning Algorithm

A behavioral cloning algorithm [10] is a representative imitation learning algorithm [25]. The behavioral cloning algorithm collects states and actions from game data collected by experts and uses them as learning data. Based on the collected data, it is possible to train an AI that acts similarly to the policy of the expert. However, behavioral cloning algorithms improve performance only in states included in the game data of experts. Ross et al. [26] improved a behavioral cloning algorithm by adjusting the ratio of the actions of an expert and other actions. Goecks et al. [9] introduced a method to simultaneously solve the problems of imitation learning and reinforcement learning.

3.4. Comparison of Proposed Method to Related Works

Jia et al. [7] performed multi-agent reinforcement learning, position learning, episode classification learning, and curriculum learning for a basketball game. The offense episode of a shooting guard in a basketball game has 42 actions. The multi-agent reinforcement learning assigned rewards to successful actions, including score, block, mark, steal, and pick. The episode-classification learning classified offense episodes into three categories: attack, assist, and ball clear.

The proposed method defines the offense episodes into 11 actions: movement in eight directions, shoot, breakthrough, and pass. Since the number of actions is reduced, the learning space is reduced, and learning performance can be improved in the same episode. The proposed method gives rewards in situations, including mark and avoid, in addition to completed actions, including shoot and breakthrough. In episode classification learning, the entire episode is classified into offense, defense, and loose ball. The offense episode is not segmented for correlation considerations.

Goecks et al. [9] explained the algorithm that adds imitation learning to reinforcement learning. Learning is performed with state, action, and reward data collected from an expert and exploration by reinforcement learning AI. The proposed method uses predefined policies without directly applying the state, action, and reward collected from an expert; it uses the state value, not the action value.

Table 1 summarizes the key differences and similarities between our work and the others. The contents in black illustrate the similarities, and items in red illustrate the differences.

Table 1. Summarizing key differences and similarities.

	Ours	Jia et al. [7]	Goecks et al. [9]
State	Normalized relative values based on basketball game data	Basketball game data	Various values
Reward	Values based on state and action	Values based on success of the action	Conventional values
Episode	Offense, defense, and loose ball	Attack, assist, defense, free ball, and ball clear	Single episode
Algorithm	Behavioral-cloning-based A2C	Multi-agent reinforcement learning	Reinforcement learning that combines imitation learning

4. Behavioral-Cloning-Based A2C and Application in Basketball Games

This section proposes a behavioral-cloning-based A2C and the method to apply it to a basketball game. The loss function for learning by behavioral-cloning-based A2C using the policies of an expert is explained. Furthermore, this section describes the structure used to apply behavioral-cloning-based A2C to a basketball game and the series of pre-processes, which include state representation, reward function, and episode classification.

4.1. Behavioral-Cloning-Based A2C

We propose a method to reduce the learning time of A2C using the behavioral-cloning algorithm [10] when the policy of an expert is provided. In general, the total loss function $\mathcal{L}^{a2c} = \mathbb{E}_{s,a\sim\pi_\theta}\left[\mathcal{L}^{policy} + \mathcal{L}^{value}\right]$ [12] is calculated using the sum of the policy loss function $\mathcal{L}^{policy} = -\log \pi_\theta(a|s)(R - V_\theta(s)) - \alpha\mathcal{H}^{\pi_\theta}$ [12] and the value loss function $\mathcal{L}^{value} = \frac{1}{2}(R - V_\theta(s))^2$ [12]. The entropy loss function is defined as $\mathcal{H}^{\pi_\theta} = -\sum_a \pi_\theta(a|s)\log \pi_\theta(a|s)$ [12]. The entropy loss function prevents the soft-max distribution from exceeding a certain range. We propose a policy loss function $\mathcal{L}^{policy}$ as specified in Equation (1) below. Where $-\log \pi_\theta(a|s)(R - V_\theta(s)) - \alpha\mathcal{H}^{\pi_\theta}$ is the same as the policy loss function of A2C $\mathcal{L}^{policy}$, and $-\beta \log \pi_\theta(a^{expert}|s)$ is the behavioral cloning loss function. When the action of an expert is determined as $a^{expert} \sim \pi^{expert}(a^{expert}|s)$, the loss of $\pi_\theta(a^{expert}|s)$ is calculated using a log function.

$$\mathcal{L}^{policy} = -\log \pi_\theta(a|s)(R - V_\theta(s)) - \alpha\mathcal{H}^{\pi_\theta} - \beta \log \pi_\theta\left(a^{expert}|s\right) \quad (1)$$

Algorithm 1 is the pseudo-code for the proposed method. Learning is repeated until the return converges. The action a_t is calculated at π_θ using the state s_t, and the action of the expert a_t^{expert} is calculated at π^{expert} using the state s_t. The reward r_t is determined based on the state s_t and action a_t. The action a_t, action of the expert a_t^{expert}, and reward r_t are written on the update buffer $\mathcal{E}$. When the episodes end, the parameter θ of the actor and parameter θ^v of the critic are updated as explained below. The return R_t is calculated by multiplying the reward r_k with gamma γ^{k-t}. The parameter θ of the actor is updated using Equation (1). The parameter θ^v of the critic is updated using the value loss function $\mathcal{L}^{value}$. When the update for one episode is completed, the update buffer is initialized.

Algorithm 1 Proposed behavioral-cloning-based A2C.

Initialize parameter θ, θ^v
Initialize update buffer $\mathcal{E} \leftarrow \varnothing$
When the return does not converge, do
  $r_t \leftarrow$ Execute an action $a_t \sim \pi_\theta(a_t|s_t)$
  Get expert action $a_t^{expert} \sim \pi^{expert}\left(a_t^{expert}|s_t\right)$
  Store transition $\mathcal{E} \leftarrow \mathcal{E} \cup \{s_t, a_t, a_t^{expert}, r_t\}$
  If the episode is done
    Compute return $R_t = \Sigma_{k=1}^{\infty}\gamma^{k-t}r_k$ for all t in $\mathcal{E}$
    # Perform actor update
    $\theta \leftarrow \theta - \pi_\theta(a|s)(R - V_{\theta^v}(s)) - \alpha\mathcal{H}^{\pi_\theta} - \beta \log \pi_\theta\left(a^{expert}|s\right)$
    # Perform critic update
    $\theta^v \leftarrow \theta^v + \frac{1}{2}(R - V_{\theta^v}(s))^2$
    Clear update buffer $\mathcal{E} \leftarrow \varnothing$
  End If
End While

4.2. Application of Behavioral-Cloning-Based A2C

We applied the proposed method to the Freestyle learning simulator environment for a 3 to 3 basketball game (hereinafter referred to as "basketball game"). The learning simulator had the same game rules as the basketball game shown in Figure 1 and provided the speed control, AI command interface, and learning data. Figure 1 was extracted from the basketball game, Freestyle (Joycity Corp, Seongnam-si, Republic of Korea).

Figure 1. Freestyle basketball game screen (Freestyle, Windows PC, Joycity Corp. 2004).

The basketball court was a half-court, which was 15 m wide and 11 m long in the virtual environment. Foul regulations such as outs or double dribbles were not applied. The ball could not go out of the court. When the offensive team failed at offense within 20 s, it was considered a foul, and the offense's turn went to the opposing team. Each team had three players. At the end of the game, the team with the higher scores won.

Figure 2 describes the behavioral-cloning-based A2C structure proposed in this paper. An agent is a basketball game character determining an action after receiving the state from the environment of the learning simulator. The actions are movement in eight directions, shoot, pass, mark, steal, rebound, and breakthrough.

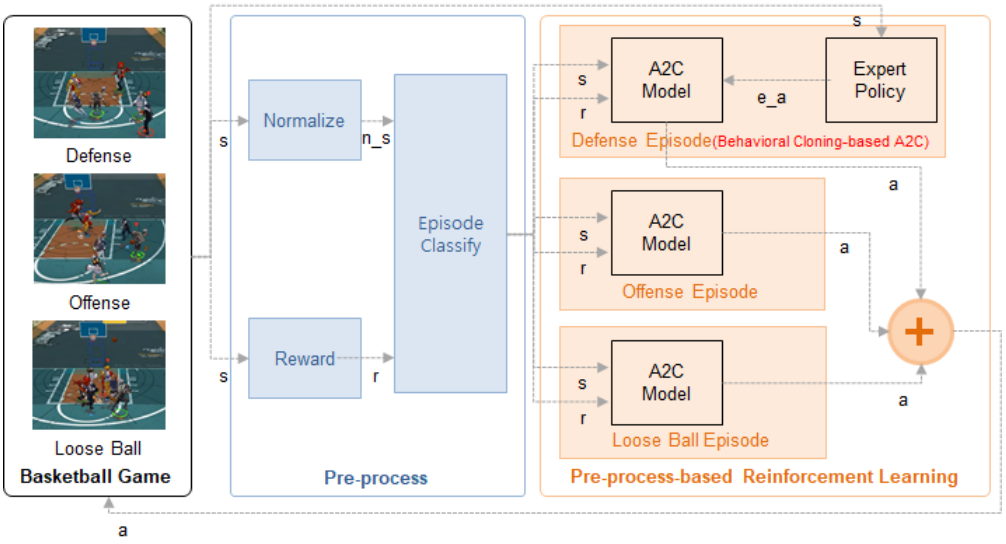

Figure 2. Structure of proposed method.

The normalization module in the basketball game receives only the state and normalizes it. The state s_t includes the location, angle, distance, conditions, time, and points in a basketball game and is expressed as shown in Table 2. Team(AI, n) means the nth player in the AI team. Enemy(AI, m) means the mth player in the opposing team. Mark(AI) means the AI's target player.

Table 2. A2C basketball game state.

State	Equation
AI location	$\frac{-z(AI)}{7.5}, \left(\left(\frac{x(AI)-1}{11}\right)-0.5\right)\times 2$
Team location	$\frac{z(AI)-z(team(AI,\,n))}{15}, \frac{x(team(AI,n))-x(AI)}{11}$
Enemy location	$\frac{z(AI)-z(enemy(AI,\,m))}{15}, \frac{y(enemy(AI,\,m))-x(AI)}{11}$
Ball location	$\frac{z(AI)-z(ball)}{15}, \frac{x(ball)-x(AI)}{11}$
Ball height	$\min\left(1.0, \frac{y(ball)}{5}\right)$
Rim location	$\frac{z(AI)}{7.5}, \frac{10.5-x(AI)}{9.5}$
Team angle	$\frac{\arccos\left(\frac{vector(rim,\,AI)\cdot vector(team(AI,\,m),\,AI)}{\lvert vector(rim,\,AI)\rvert\times\lvert vector(team(AI,\,m),\,AI)\rvert}\right)}{\pi}$
Enemy angle	$\frac{\arccos\left(\frac{vector(rim,\,AI)\cdot vector(enemy(AI,\,m),\,AI)}{\lvert vector(rim,\,AI)\rvert\times\lvert vector(enemy(AI,\,m),\,AI)\rvert}\right)}{\pi}$
Mark angle	$\frac{\arccos\left(\frac{vector(rim,\,mark(AI))\cdot vector(AI,\,mark(AI))}{\lvert vector(rim,\,mark(AI))\rvert\times\lvert vector(AI,\,mark(AI)\rvert}\right)}{\pi}$
Ball distance	$\min\left(1.0, \frac{dist(ball,\,AI)}{10}\right)$
Rim distance	$\min\left(1.0, \frac{dist(rim,\,AI)}{10}\right)$
Whether ball is clear or not	$ball\_clear(AI)$
Whether pivoting or not	$pivot(AI)$
Game time remaining	$\frac{quarter\_time}{240}$
Foul time remaining	$\frac{violation\_time}{20}$
Team scores	$\min\left(1.0, \frac{score(AI)}{30}\right)$

Location means the location of a character, ball, and rim. Each character took their location as the origin and expressed other characters, the ball, and the rim as relative locations. Figure 3 illustrates the normalization process. The blue dot is the reference character, and the red dots are other objects. The AI location uses the relative value around the center of a court. The location is expressed in 2D, but in the case of a ball, height is considered. The maximum height of a ball is 5 m, with zero (0) as the basic level.

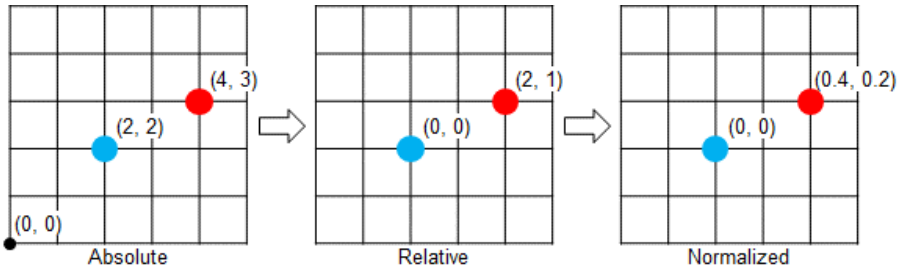

Figure 3. AI location normalization process in a basketball game.

The angle is not state-provided in the environment but is calculated and used as a state value. It is used as the reference value to identify whether the character is open. The distance is calculated by dividing the distance between the reference object and another object by the maximum allowable distance (10 m). It uses the unit meter. The conditions are whether the AI team clears the ball (ball_clear(AI)) and whether the AI pivots (pivot(AI)). The conditions have the values 0 or 1.

There are two kinds of time in seconds, the time indicating the game time remaining until the game finishes and the foul time. These are divided into a maximum of 240 s and 20 s, respectively, and normalized to values between 0 and 1. The scores display the current scores of the AI team, and the value divided by the maximum allowable scores (30 points) was used as the state.

The reward module in the basketball game established a reward system to apply A2C. The reward r_t was defined as a shoot, pass, breakthrough, mark, avoid, etc., as shown in Table 3, and is calculated by adding all rewards generated. The importance of the frequency of the actions breakthrough, mark, avoid, and pick is considered when the corresponding rewards are calculated.

Table 3. A2C basketball game reward.

Rewards	Equation
Shoot	Open status $(0.0/1.0)\times$ Within the range $(0.0/1.0)\times$ Shoot status $(0.0/1.0)$
Pass	Pivot status $(0.0/1.0)\times$ Pass status $(0.0/1.0)$
Breakthrough	Within the range $(0.0/1.0)\times$ Breakthrough status $(0.0/1.0)\times$ Weighted value (0.5)
Mark	Mark angle ≤ 30 $(0.0/1.0)\times$ Mark distance $\leq 2(0.0/1.0)\times$ Weighted value (0.002)
Avoid	Open status $(0.0/1.0)\times$ Within the range $(0.0/1.0)\times$ Weighted value (0.01)
Ball clear	Ball clear status $(0.0/1.0)$
Pick	Pick status $(0.0/1.0)-$ Movement away from a ball $(0.0/1.0)\times$ Weighted value (0.002)
Under rim	Rim distance ≤ 2 m $(0.0/1.0)\times$ Weighted value (0.001)

The episode classification module classifies the state s_t, action a_t, and reward r_t into the offense, loose ball, and defense situation and sends them to the corresponding A2C module. The episode classification criteria depend on the ball possession status. When a team has the ball, it is an offense episode. When an enemy has the ball, it is a defense episode. When no team has the ball, it is a loose ball episode.

According to the episode classification results, the state, reward, and action used in learning depend on the offense, loose ball, and defense situation. To control the characters by the situation, A2C was applied to offense and loose ball episodes, and the proposed method was applied to defense episodes. The defense episode is a situation in which the enemy has the ball, and the AI team needs to be within ±30 degrees from the marked angle and within 2 m from the mark distance, as specified in the rewards in Table 2. The behavioral-cloning-based A2C learning designs expert policies through distance and coordinates calculations and imitates expert actions. The proposed method, applied to defense, can also be applied to offense and loose ball situations.

5. Experiment

The proposed behavioral-cloning-based A2C was experimentally evaluated. This section compares the performance of the proposed algorithm with the conventional A2C algorithm and analyzes the experimental results. Moreover, this section analyzes the experimental results from the conventional FSM-based AI match experiment and the expert-designed FSM-based AI match experiment.

The architecture used for learning consists of two networks with dense layers with 128 units as the actor and critic. A rectified linear activation function was used as an activation function. The output layer applies a soft-max function for determining the action and a linear function for estimating the state value in each network. All values were initialized with the default parameters defined in TensorFlow.

The environment used for learning was a basketball game simulator. The simulator provided an accelerated learning and multi-client learning environment and provided all information about the basketball game per each frame. The agent that communicated with the environment consisted of one or more models concurrently. Many tasks covering the entire learning process, such as determining learning algorithms and defining models, were handled here. Briefly, in Figure 2, all parts except the basketball game box were actually processed inside this agent. Therefore, our proposed method was also carried out in the agent. The data received from the simulator were normalized by each state equation. We were able to reduce the complexity of basketball game states through state normalization. At this time, the reward was also calculated by each equation. The calculated state and reward went through an episode classification process that was classified according to the ball's ownership. In the reduced and classified state, the model had repeatedly experienced similar states. Therefore, the model could experience various states even with episode-level online learning and offline learning rather than hindered learning. If an expert policy was available, the state and expert action were stored to calculate additional loss when updating the model. This agent was also used in experiments for verification after all learning had ended. The agent loaded the saved model after the learning was completed, conducted performance experiments with the opponent through the match, and stored the results as logs.

5.1. Behavioral-Cloning-Based A2C Experiment

During the experiment, the model learned the offense episode without the policy of an expert and the defense episode reflecting the policy of an expert, and the number of rewards per episode was measured. A total of 400,000 actions performed by the AI were consecutively collected from all episodes, and the number of rewards for the actions avoid and mark were recorded after every 1000 actions.

Figure 4 shows the A2C learning experiment results, applying the proposed behavioral cloning. Figure 4a shows the number of rewards for the avoid action, and Figure 4b shows the number of rewards for the mark action.

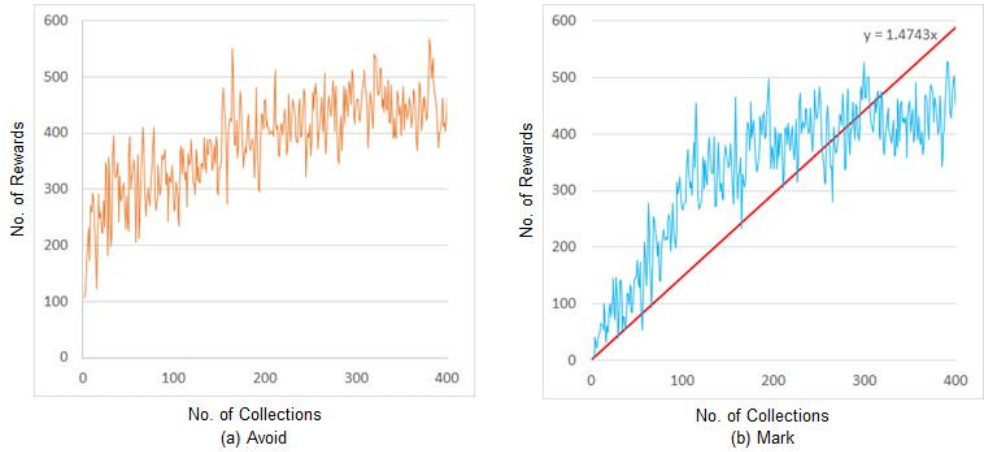

Figure 4. Experimental results of the proposed method.

Figure 5 illustrates the experimental results of the conventional A2C learning, which did not adopt behavioral cloning. Figure 5a shows the number of rewards for the avoid action, and Figure 5b shows the number of rewards for the mark action.

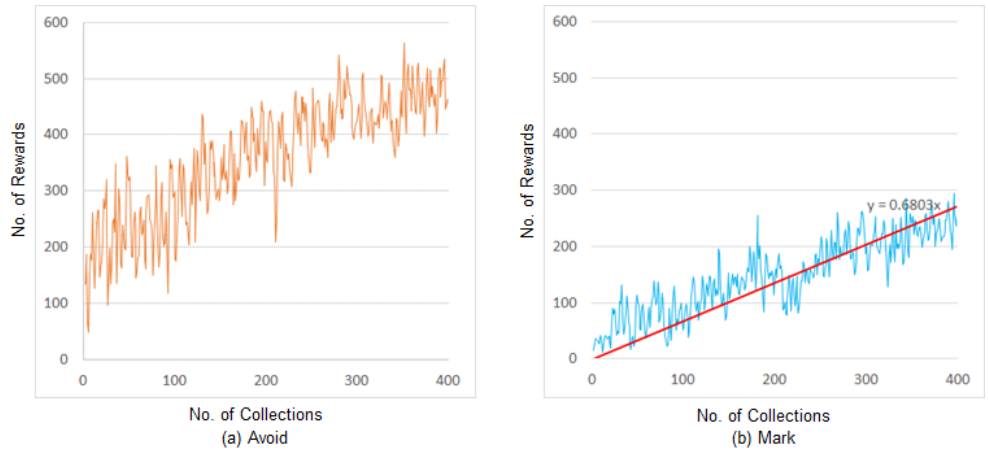

Figure 5. Conventional A2C experiment results.

The reward for the avoid action in the offense episode without the policy of the expert is similar in both of the above methods. For the defense episode with behavioral cloning and the policy of the expert, the average number of rewards in the final 100 data counts was 224 that of the conventional A2C experiment, and that for the proposed method was 428, showing its marked success, achieving values approximately 1.9 times higher than the conventional A2C.

The experimental results of the proposed method and the conventional A2C were compared using the linear trend curve with zero as the reference point because of the unstable number of avoid and mark actions. The red line in Figure 4b is the trend curve showing the number of rewards for the mark action in the experiment with the proposed method. The red line in Figure 5b is the trend curve showing the number of rewards for the mark action in the experiment with the conventional A2C. The slope of the proposed method is approximately 1.47, compared with 0.68 for the conventional A2C method. The slope of the proposed method is approximately 2.1 times higher than that of the conventional A2C. This implies that the proposed method selected and performed the actions, generating relatively more rewards than the conventional A2C.

5.2. Conventional FSM-Based AI Match Experimental Results and Analysis

Figure 6 presents the learning experiment results of the match with conventional FSM-based AI. Figure 6a shows the scores of the proposed method, and Figure 6b shows the scores of the conventional FSM-based AI. Figure 6c displays the accumulated number of wins, draws, and losses. These are the results of accumulating the number of wins (+1), draws (+0), and losses (−1) for the behavioral-cloning-based A2C AI from the match results. Figure 6d shows the total rewards per match acquired by AI in the proposed method of learning.

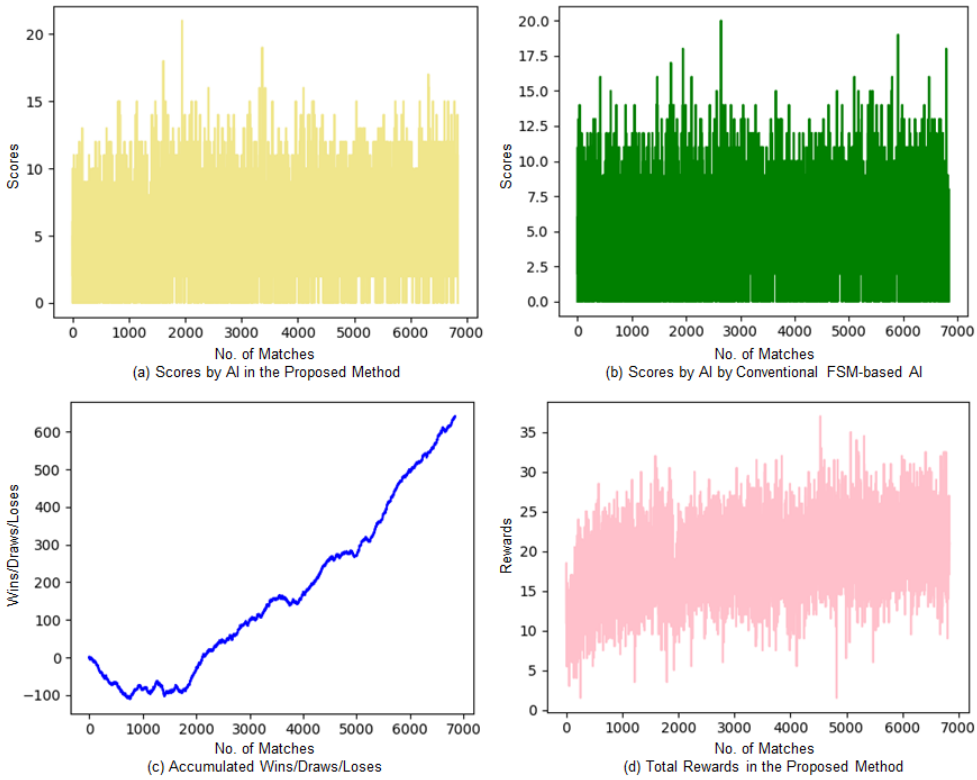

Figure 6. Experimental results for the conventional FSM-based AI match.

The proposed method shows relatively higher scores; the total average score difference is about 0.43. The average gap of scores in the matches from the first to the 1000th match with the FSM-based AI had a higher winning rate of approximately 0.4, which is higher than that of the proposed method. The proposed method achieved a higher score of approximately 0.83 in the average gap of scores in the final 1000 matches, in which the AI in the proposed method showed a higher winning rate. For accumulated wins/draws/losses, the proposed method had a relatively higher number of losses in the early stage of learning. As learning continued, the AI in the proposed method gradually reduced losses and increased the winning rate. The total rewards also converged into a constant value after gradually increasing from lower rewards at the beginning. For the final 1000 matches, the proposed method achieved a winning rate of 60%, with 512 wins, 146 draws, and 342 losses.

5.3. Expert-Designed FSM-Based AI Match Experimental Results and Analysis

The experimental match was between the AI of the proposed method and the expert-designed FSM-based AI. Figure 7 shows the match's experimental results. A total of 103 experimental matches were played. The proposed method achieved a winning rate of 66%, with 68 wins and 35 losses.

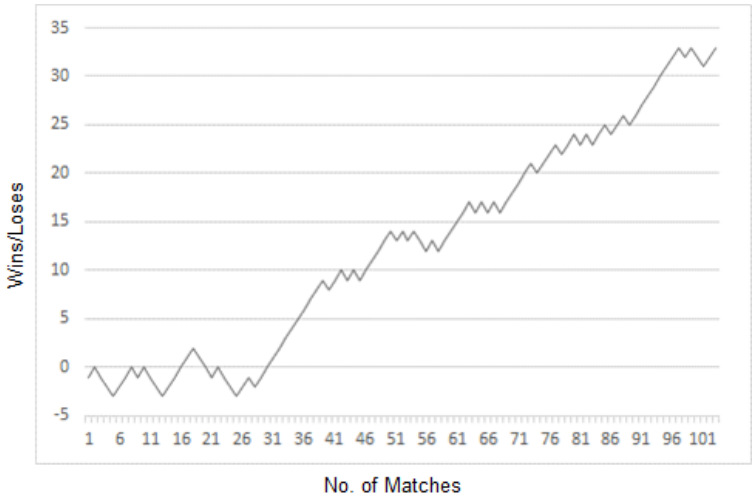

Figure 7. Accumulated wins/losses in the matches with expert-designed FSM-based AI.

The experimental results can be analyzed using the number of successful actions by a character as well as wins/losses or scores. Table 4 illustrates the average number of successful actions per match for each AI. The AI in the proposed method achieved relatively higher numbers in offense episodes (2 points) and defense episodes (block and steal) than the expert-designed FSM-based AI. The expert-designed FSM-based AI showed a higher number of rebounds than the AI in the proposed method. The AI in the proposed method can be improved through learning in loose ball episodes, including the timing for the rebounds and preoccupying positions.

Table 4. Average number of successful actions per match with expert-designed FSM-based AI.

Team	2 Points	3 Points	Rebound	Block	Steal
FSM AI	5.3	0.06	7.2	0.68	0.38
Proposed AI	6.76	0	2.4	5.3	2.33

6. Conclusions

This paper proposed a behavioral-cloning-based A2C performing actions that reflect the policy of an expert by applying a behavioral-cloning algorithm to A2C in a basketball game. The state was normalized to apply the proposed method to a complicated basketball game, and the reward function was proposed based on the state generated from matches. Learning was performed by classifying various episodes to solve the learning time challenge of reinforcement learning. The proposed method improved performance by approximately twice the performance of the conventional A2C method.

The winning rate of the proposed method was approximately 60% in the matches against the conventional FSM-based AI and approximately 66% in the matches against the expert-designed FSM-based AI. The proposed method showed a relatively higher number of successes in 2 points, steal, and block than the expert-designed FSM-based AI. However, the proposed method demonstrated a relatively lower number of successes in 3 points and rebound actions.

In future research, we plan to investigate the optimal basketball game learning algorithm by enhancing the proposed A2C algorithm. A human-level AI will be developed by analyzing the data collected from players and using it to improve the AI. Through such efforts, future research can develop a method that makes players more interested in the game.

Author Contributions: Conceptualization, T.C.; Funding acquisition, K.C.; Investigation, T.C.; Methodology, Y.S.; Project administration, K.C.; Supervision, K.C.; Validation, T.C. and Y.S. All

authors will be informed about each step of manuscript processing including submission, revision, revision reminder, etc. via emails from our system or assigned Assistant Editor. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Culture, Sports and Tourism R&D Program through the Korea Creative Content Agency grant funded by the Ministry of Culture, Sports and Tourism in 2022 (Project Name: Education & research group for advanced AI technology in the field of sports games, Project Number: R2022020003, Contribution Rate: 100%).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. *Nature* **2019**, *575*, 350–354. [CrossRef] [PubMed]
2. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-Level Control through Deep Reinforcement Learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
3. Bellemare, M.G.; Naddaf, Y.; Veness, J.; Bowling, M. The Arcade Learning Environment: An Evaluation Platform for General Agents. *J. Artif. Intell. Res.* **2013**, *47*, 253–279. [CrossRef]
4. Watkins, C.J.C.H.; Dayan, P. Q-Learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
5. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML'15), Lille, France, 6–11 July 2015; Volume 37, pp. 448–456.
6. Shao, K.; Zhu, Y.; Zhao, D. StarCraft Micromanagement with Reinforcement Learning and Curriculum Transfer Learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **2019**, *3*, 73–84. [CrossRef]
7. Jia, H.; Hu, Y.; Chen, Y.; Ren, C.; Lv, T.; Fan, C.; Zhang, C. Fever Basketball: A Complex, Flexible, and Asynchronized Sports Game Environment for Multi-Agent Reinforcement Learning. *arXiv* **2020**, arXiv:2012.03204.
8. Ng, A.Y.; Harada, D.; Russell, S. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In Proceedings of the Sixteenth International Conference on Machine Learning (ICML '99), Bled, Slovenia, 27–30 June 1999; pp. 278–287.
9. Goecks, V.G.; Gremillion, G.M.; Lawhern, V.J.; Valasek, J.; Waytowich, N.R. Integrating Behavior Cloning and Reinforcement Learning for Improved Performance in Dense and Sparse Reward Environments. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS'20), Auckland, New Zealand, 9–13 May 2020; pp. 465–473.
10. Bain, M.; Sammut, C. A Framework for Behavioural Cloning. *Mach. Intell.* **1995**, *15*, 103–129.
11. Konda, V.; Tsitsiklis, J. Actor-Critic Algorithms. In Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS'99), Cambridge, MA, USA, 29 November–4 December 1999.
12. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.P.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML'16), New York, NY, USA, 19–24 June 2016; Volume 48, pp. 1928–1937.
13. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. In Proceedings of the International Conference on Learning Representations (ICLR), San Juan, PR, USA, 2–4 May 2016.
14. Hasselt, H.V.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double Q-Learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16), Phoenix, AZ, USA, 12–17 February 2016; pp. 2094–2100.
15. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.V.; Lanctot, M.; Freitas, N.D. Dueling Network Architectures for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML'16), New York, NY, USA, 19–24 June 2016; Volume 48, pp. 1995–2003.
16. Bellemare, M.G.; Dabney, W.; Munos, R. A Distributional Perspective on Reinforcement Learning. In Proceedings of the 34th International Conference on Machine Learning (ICML'17), Sydney, Australia, 6–11 August 2017; Volume 70, pp. 449–458.
17. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
18. Oh, J.; Guo, Y.; Singh, S.; Lee, H. Self-Imitation Learning. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 3878–3887.
19. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540.

20. Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A.S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J.; Schrittwieser, J.; et al. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv* **2017**, arXiv:1708.04782.
21. Samvelyan, M.; Rashid, T.; Witt, C.S.D.; Farquhar, G.; Nardelli, N.; Rudner, T.G.J.; Hung, C.M.; Torr, P.H.S.; Foerster, J.; Whiteson, S. The StarCraft Multi-Agent Challenge. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS'19), Montreal, QC, Canada, 13–17 May 2019; pp. 2186–2188.
22. Kurach, K.; Raichuk, A.; Stańczyk, P.; Zając, M.; Bachem, O.; Espeholt, L.; Riquelme, C.; Vincent, D.; Michalski, M.; Bousquet, O.; et al. Google Research Football: A Novel Reinforcement Learning Environment. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 4501–4510. [CrossRef]
23. Liu, Y.; Wu, F.; Lyu, C.; Li, S.; Ye, J.; Qu, X. Deep dispatching: A deep reinforcement learning approach for vehicle dispatching on online ride-hailing platform. In Proceedings of the 4th International Symposium on Multimodal Transportation (ISMT'21), Nanjing, China, 14–15 December 2021. [CrossRef]
24. Peng, X.B.; Panne, M.V.D. Learning Locomotion Skills Using DeepRL: Does the Choice of Action Space Matter. In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'17), Los Angeles, CA, USA, 28–30 July 2017; pp. 1–13.
25. Osa, T.; Pajarinen, J.; Neumann, G.; Bagnell, J.A.; Abbeel, P.; Peters, J. An Algorithmic Perspective on Imitation Learning. *Found. Trends Robot.* **2018**, *7*, 1–179. [CrossRef]
26. Ross, S.; Gordon, G.J.; Bagnell, J.A. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS), Ft. Lauderdale, FL, USA, 11–13 April 2011.