A Memory Driven Self-learning Combat Agent Architecture in a 3D Virtual Environment

Tianci Zhang[1,*], Yongyong Wei[2] and Hao Fang[1]

[1]*School of Automation, Beijing Institution of Technology, Beijing 100081, China*
[2]*School of Astronautics, Beijing Institution of Technology, Beijing 100081, China*
E-mail: tomandjames8899@outlook.com
[*]*Corresponding Author*

Received 24 March 2025; Accepted 10 April 2025

Abstract

Agent behavior modeling in 3D virtual environments is a critical challenge in artificial intelligence and military simulation. While rule-based methods (e.g., finite state machines) are widely used, their limitations in adaptability and development efficiency hinder their application in dynamic combat scenarios. To address this, a memory-driven self-learning agent (MDSLA) architecture is proposed, integrating visual, auditory, and game features to simulate human-like battlefield decision-making. The architecture employs an asynchronous advantage actor-critic (A3C) framework to enhance training efficiency and incorporates a memory module for processing historical perception data. Experimental validation in the Vizdoom environment demonstrates that MDSLA outperforms traditional rule-based methods and mainstream reinforcement learning algorithms in convergence speed and combat effectiveness. Furthermore, a parallel simulation mechanism is implemented via high-speed middleware, enabling seamless deployment of the model on both Vizdoom and a high-precision simulation platform

Journal of Web Engineering, Vol. 24_5, 687–712.
doi: 10.13052/jwe1540-9589.2451

(HPSP). Results from HPSP experiments show a 33% reduction in task execution time and a 24.1% improvement in lethality compared to finite state machine-driven agents. This work provides a scalable framework for developing intelligent combat agents with enhanced adaptability and realism in 3D virtual environments.

Keywords: Agent modelling, memory-driven architecture, reinforcement learning, military simulation, 3D virtual environment.

1 Introduction

When designing military simulation applications or developing high quality video games, the most difficult and important part is how to make our agents and opponents smarter and robust and easy to implement while providing high illusion of immersion [1]. In particular, in games or applications that players and its opponents conflict directly with each other using weapons in a given scenario until either side is defeated or killed [2], in this paper, this kind of problem is described as a "death matching" game. In traditional applications, to ensure fluency and antagonism of the game, agents are always fed with existing rules or experiences and other prior knowledge [3]. However, it is well-known that a rule based agent always behaves as mechanized and is easy to defeat in virtual environments [4]. Developing an agent model with rules is extremely time-consuming. Developers first need to code rules into scripts and then need to debug them to make sure their compatibility in a virtual environment [5]. To handle this problem in both intelligence and efficiency, it would be more effective to use a method that does not depend on prior knowledge and could evolve by itself to find optimal action under certain circumstances [6]. Agent behavior modeling is a research area where the agent make actions under certain combat circumstances [7]. There are several research areas, such as agent path planning, agent decision making, multi-agent coordination behavior modeling, and others [8]. For instance, single agent decision making is an essential task for a combat agent that meet its enemies when fighting in a complex scenario [9]. This task is known as the death matching agent modeling (DMAM) problem. A deathmatch game is a game mode in confrontation games where the ultimate goal is to maximize the number of kills by a player/agent while minimizing casualties and consumables, it could be 1 vs. 1 or multi vs. multi [10–12]. It is composed of both single agent path planning and action selecting [13]. Reinforcement learning has been favored in dealing with DMAM in a virtual environment [14].

Recent works have utilized both imitation learning (IL) and reinforcement learning (RL) to improve agent performance [15–17]. For IL, teacher forcing has been utilized for training the agent, whereas online policy learning is used for training the agent in RL. And yet, it is difficult for agents to make rational and timely decisions in certain deathmatching scenarios. Considering the scenario where red agents and blue agents are in the same virtual environment, armed with weapons and ammo, both wanting to defeat the other. Both sides need to make proper decisions to win deathmatching games. Such tasks face even more difficulty in a 3D simulation environment [18].

To address this problem, we propose a visual-hearing-neuro logic network, which has the ability to utilize visual and hearing clues as the perception of agents and fuse them with pixels and sound wave features with convolution and sound recognition. Moreover, a neuro logic network based on an A3C (asynchronous advantage actor critical) architecture to generate optimal policies is added to simulate the memory and logic processing of a combating agent, which utilizes multiple parallel actor-learners to asynchronously update the global model parameters, significantly accelerating training convergence by reducing sample correlation and enhancing exploration efficiency.

This study outlines an end-to-end framework that comprised three modules: a module of feature processing components for visual and hearing, a neuro logic processing module that processes both agent memory and current input with game features, and an A3C training network that accelerates training speed that helps the agent in deciding the best action to make [19]. The main contributions of this paper are as follows:

(1) An agent behavior modeling architecture using a first person view and sound information along with game features as input features is proposed, which could simulate agent memory both in visual and hearing. The model is tested in the Vizdoom environment and results demonstrate that our model outperforms traditional models both in time and reward.
(2) The model architecture is integrated with the A3C network, enabling it to simultaneously process the agent's memory information and current information during operation. By leveraging the A3C architecture, its training efficiency is significantly improved. Additionally, the overall game features and reward function is designed and simulation verification is conducted.
(3) The effectiveness of the proposed model in real applications is verified by using a high precision simulation platform (HPSP) which

provide realistic 3D interaction and high-confidence physical computation capabilities. Simulation scenarios is built using same structure and coordinates as in the Vizdoom environment and a parallel simulation deduction experiment was conducted through high speed middleware. Its performance is compared with agents using rule-based finite state machines.

2 Related Work

Agent behavior modeling has been a hot spot both in the video games industry and AI research community in recent years [20], and three major approaches exist in the literature: the finite state machine method, behavior tree method and reinforcement learning method. Although the finite state machine method has been widely used in many successful military simulation platforms or video games such as Virtual Battlespace or Arma 3 [21], it has limitations both on the number of states and parallel processing capability, and it is hard to optimize performance over different scenarios. The behavior tree method provides more scalability to constantly changing needs and environments and flexibility to adjust parameter and optimize policy, but it still depends on sufficient training data and is prone to overfitting due to the hierarchical structure and complexity of the behavior tree [22]. The deep reinforcement learning method is goal oriented; DRL agents are able to learn and adjust behavioral strategies base on predetermined goals to achieve goal optimization without training data, and have strong adaptability on environmental changes and generalization ability from one task to other similar tasks [23]. To achieve the goal of building a self-motivated and life-like combat agent, the DRL method has to be combined with image processing and game feature processing and a fuse network [24]. The next section describes related works for DRL agent behavior modeling.

2.1 Agent Behavior Modeling in a 2D Environment

In 2013, DeepMind presented the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning, and applied the method to seven Atari 2600 games from the Arcade Learning Environment; the method proved its state of art by outperforming all previous approaches and surpassing a human expert on three of them [25]. On 13 April 2019, OpenAI Five became the first AI system to defeat world champions at an esports game [26]. The game

of Dota 2 presents novel challenges for AI systems such as long time horizons, imperfect information, and complex, continuous state-action spaces, all challenges which will become increasingly central to more capable AI systems. By defeating the Dota 2 world champion (Team OG), OpenAI Five has demonstrated that on a difficult task, self-play reinforcement learning can also achieve superhuman performance. An agent from DeepMind, AlphaStar, has also achieved grandmaster level performance by beating top professional players in the most popular real-time strategy game StarCraft II, currently played worldwide [27].

2.2 Agent Behavior Modeling in a 3D Environment

Although recent advances in deep neural networks have led to effective vision-based reinforcement learning methods that have been employed to obtain human-level controllers in Atari 2600 games, Dota 2 and Starcraft from pixel data. These games, however, do not resemble real-world tasks since they involve non-realistic 2D environments and the third-person perspective. In 3D simulation applications and video games, as noticed, the most widely used agent behavior modeling method is always traditional methods such as FSM and behavior tree. In Virtual Battlespace, the state of art military simulation 3D training system, developed by Bohemia interactive, uses FSM to simulate basic behavior such as move, jump or patrolling which allows developers to build more complex tactical action using a behavior tree [28]. These two methods have gained huge success in military applications all over the world. However, their generalization to scenario and restriction in modeling efficiency is a fatal drawback to build agile and fast-responding agents. Hence, researchers are seeking a way to build a self-learning agent behavior modeling method in a 3D environment such as VBS. Pozan University of Technology propose a novel test-bed platform for reinforcement learning research from raw visual information which employs the first person perspective in a semi-realistic 3D world [29]. The software, called Vizdoom, is based on the classical first-person shooter video game, Doom. It allows developing bots that play the game using the screen buffer. The birth of the Vizdoom platform was of huge help to a related researcher, Guillaume Lample, from Carnegie Mellon University who proposed an AI-agent for playing death-matches in FPS games using only the pixels on the screen [30]. The research team evaluated the model on the two different tasks under the Vizdoom environment. The Unity and Unreal engines also provided a machine learning developing toolkit for researchers to test their novel agent behavior models in 3D environments [31, 32].

2.3 Policy Output Network

Guillaume Lample divides the death-matching problem into two phases, navigation and action, and uses separate networks for each phase of the game. Furthermore, the agent infers high-level game information, such as the presence of enemies on the screen, to decide its current phase and to improve its performance. The proposed architecture has been proved to substantially outperform built-in AI agents of the game as well as humans in deathmatch scenarios in the Vizdoom virtual environment. Adil Khan also trained a neural network with simple architecture using a direct Future prediction method under a Vizdoom 3D environment; the agents performed well against both human players and inbuilt game agents [33]. Khan Adil and Shaohui Liu proposed a method that uses convolutional deep learning with a Q-learning algorithm trained on Doom's basic scenario(s) in the same semi-realistic 3D environment as Vizdoom, which considers only the screen's raw pixels for exhibiting an agent's usefulness. Shashank Hegde from University of Southern California also proposed a new approach that uses both visual images and audio buffers to train combat agents' policy output neural network [34].

3 Proposed Method

In this study, a novel framework for an agent in a combat scenario is proposed, which comprises a memory network and decision network, called MN+DN, that is capable of improving an agent's performance when facing deathmatching scenarios. Specifically, memory network utilizes both visual images and audio sounds as input features to produce memory and then transfer computing results to a decision network; a decision network uses both computing results from the memory module and current input information with game features to produce a final decision.

In the Vizdoom environment, our model consists of three major processing parts as shown in Figure 2. First there are two memory feature processing modules: one is for visual image input and the other is for audio input. The first person visual images of agents is used for extracting visual features, consisting of a convolution neural network, whereas an audio encoder architecture is used for extracting information about variety and relative position. Second, there is a decision processing network, which utilizes game features and memory information and instantaneous visual and audio information. Third, an actor-critic network is used to combine memory and decision network. The proposed fusion network enhances the performance of the

Table 1 Image processing network design

Layer	Input Size	Filters	Kernel	Stride	Activation
1	$84 \times 84 \times 3$	32	8×8	4	RELU
2	$20 \times 20 \times 32$	64	4×4	2	RELU
3	$9 \times 9 \times 64$	64	3×3	1	RELU

whole model greatly. Finally, a policy learning module is designed to produce the appropriate output actions and it leverages the asynchronous advantage actor critic (A3C) policy gradient method.

3.1 Feature Processing

While a soldier is executing a given combat task in the real world, they are receiving visual and audio information all the time; the agent recognizes this information and then stores it into memory. At each step during an episode, a raw-pixel image and audio vector are given to the agent. Both the image and the audio vector are processed with our memory feature processing module to produce memory features.

3.1.1 Image processing

Visual features are extracted from input images through a three convolution layers neural network. The first layer is a 2D convolution layer with 32 filters, the second and last layers are two 2D convolution layers with 64 filters. It is used as the feature extractor to transform screen pixels into a vector of length 512. The number of filters, and activation functions are shown in Table 1.

First-person visual features are produced by three layers v^1_{image}, v^2_{image}, v^3_{image} as in Equation (1).

$$v_{image} = [v^1_{image}, v^2_{image}, v^3_{image}] = CNN(I_{image}) \tag{1}$$

where I_{image} is the input image.

3.1.2 Audio processing

An audio encoder is utilized for generating a compact representation of the raw sound data. This representation is then concatenated with the features from the image processing network. The raw audio input is a vector $s \in R^n, s\_i \in [-1, 1]$ containing *n* normalized audio samples. Vizdoom runs at a fixed 35 frames per (real time) second [38], so for each simulation step this input contains audio corresponding to 29 ms of gameplay. Thus the interval

between an audio sample is too short for a real human to process while combating, so we downsize the audio samples by averaging audio samples from 35 frames in 1 second.

$$A_{\textit{per-second}} = \frac{A_{n:1} + A_{n:2} + \cdots + A_{n:35}}{35} \tag{2}$$

We then feed the samples through two 1D convolutional layers. While this removes high-frequency components (anything above ∼3000 Hz), most information lies below this frequency threshold. This down sampling allows us to reduce computational complexity. the computing results is stored in arrays allocated in memory, used as input vectors both to memory network and decision network.

With the fixed 22,050 HZ sampling rate and standard 4-frameskip, our audio observation consists of 2520 samples, or 114 ms of audio. We process both left and right audio channels separately and concatenate channels features into a single output vector. Figure 3 illustrates the high-level structure of the encoders.

3.1.3 Memory network

A novel memory network frame which could process not only video images, but also audio features is proposed. As mentioned, an agent's first person visual image is extracted by a three-layer convolutional neural network and transformed into a vector of length 512. We assume that a combat agent can hold a memory with 5 consecutive seconds. Thus the last five image feature and audio vectors corresponding to each last five seconds is stored, features as a two input features separately into two full connected neural networks is combined to produce output as memory features. Detailed algorithms are shown in Tables 2 and 3.

4 Agent Network Design

Based on the above feature processing flow, a combat decision network is proposed, which utilizes features processed by a video-audio memory process network as input; in addition, as the decision making of a combat behavior is not only affected by its memory, but also its instantaneous perception, so the image and audio from the current frame is added as features feed as input to our decision network. To ensure convergency of the whole model, we utilized game features as another model feature. These features are transferred into a

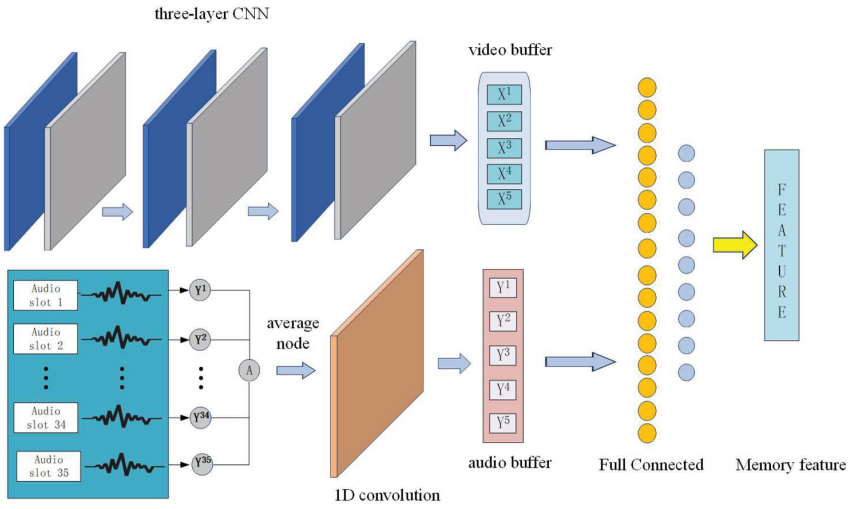

Figure 1 Memory processing network architecture.

Table 2 Training algorithm of our video-audio memory pre-processing network

Algorithm 1: Memory Pre-Processing Network Training Algorithm

1. Given an first person image dataset D_v = [x_k] and a matched audio dataset D_a = [y_k] are collected when the combat agent traverses in the environment randomly.
2. Train a three-layer convolutional neural network using D_v to obtain video features network N_v.
3. Train a 1-D convolutional neural network using D_a to obtain audio features network N_a.
4. Epoch is the number of epochs to be iterated
5. b_v is the number of images of a batch, b_a is the number of audio of a batch.
6. L_{MSE} is the Mean Square Error loss function
7. n_v is the number of images in the dataset, n_a is the number of images in the dataset
8. For k to epoch do:
9. For 1 to n by b do
10. With batch instance $x = \{x_l^{l+b}\}$, the video and audio batch instances are x_{video}, x_{audio}.
11. $x_{video} = N_v(x)$ where N_v is the three-layer CNN layers.
12. $x_{audio} = N_a(x)$ where N_a is the 1D - CNN layers.
13. $L_{MSE-video}(x, x_{video}) = \sum_l^{l+b} \sum_{i=1}^{w} \sum_{j=1}^{h} (x^{l,i,j} - x_{video}^{l,i,j})^2$
14. $L_{MSE-video}(x, x_{audio}) = \sum_l^{l+b} \sum_{i=1}^{w} \sum_{j=1}^{h} (x^{l,i,j} - x_{audio}^{l,i,j})^2$
15. Perform backpropagation to update the parameters of N_v and N_a.

Table 3 Training algorithm of our proposed model

Algorithm 2: Training Algorithm of Our Proposed Model

1. Given a pre-trained video features network $\mathrm{N_v}$ and audio features network $\mathrm{N_a}$ as in Algo. 1.
2. I^t_{image}, I^t_{audio} are input image, audio at time t, respectively.
3. x_{video}, x_{audio} are feature vectors of the input image and audio, respectively.
4. n^{buffer} is the iterator for both video and audio buffers.
5. b_v is the number of images of a batch, b_a is the number of audio of a batch.
6. Initialize thread step counter $t \leftarrow 1$, $n^{buffer} \leftarrow 0$.
7. repeat
8. Reset gradient information: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$
9. Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
10. $t_{start} = t$ and get state $s_t = [I^t_{image}, I^t_{audio}]$
11. transform s_t to pre-processing network: $x^t_{video} = N_v(I^t_{image}), x^t_{audio} = N_a(I^t_{audio})$
12. Initialize video image buffer and audio buffer into empty vector of length 5.
13. do first time repeat until $n^{buffer} = 5$
14. Insert I^t_{image} into end of the vector b_{video} and I^t_{audio} into end of the vector b_{audio}
15. $n^{buffer} \leftarrow n^{buffer} + 1$
16. Insert I^t_{image} into end of the vector b_{video} and I^t_{audio} into end of the vector b_{audio}
17. Get $b_{video} = [x^{t-4}_{video}, x^{t-3}_{video}, x^{t-2}_{video}, x^{t-1}_{video}, x^t_{video}]$, $b_{audio} = [x^{t-4}_{audio}, x^{t-3}_{audio}, x^{t-2}_{audio}, x^{t-1}_{audio}, x^t_{audio}]$
18. $f_{union} = Concatenation(f_{video}, f_{audio})$
19. $f_{union-decoder-1} = FC(f_{union}, 128)$ (Fully Connected)
20. $f_{union-decoder-1} = FC(f_{union}, Action - number)$
21. Get probability vector of actions from $f_{union-decoder-1}$
22. Select action a_t which has biggest with probability ϵ and an arbitrary action with $1 - \varepsilon$.
23. Perform a_t and receive reward r_t and new state $s_{t+1} = [I^{t+1}_{image}, I^{t+1}_{audio}]$
24. $t \leftarrow t + 1$.
26. Until terminal s_t or agent is killed by enemies.
27. for $i \in t-1, \ldots, t_{start}$ do
28. $R \leftarrow r_i + \gamma R$
29. Accumulate gradients w.r.t
30. θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} log\pi(a_i|s^{union}_i; \theta')(R - V(s^{union}_i; \theta'_v))$
31. Accumulate gradients information w.r.t
32. θ'_v: $d\theta_v \leftarrow d\theta_v + \partial(R - V(s^{union}_i; \theta'_v))^2/\partial\theta'_v$
33. Perform asynchronous update of $d\theta$ and of θ_v using $d\theta_v$
34. Until $T > T_{max}$

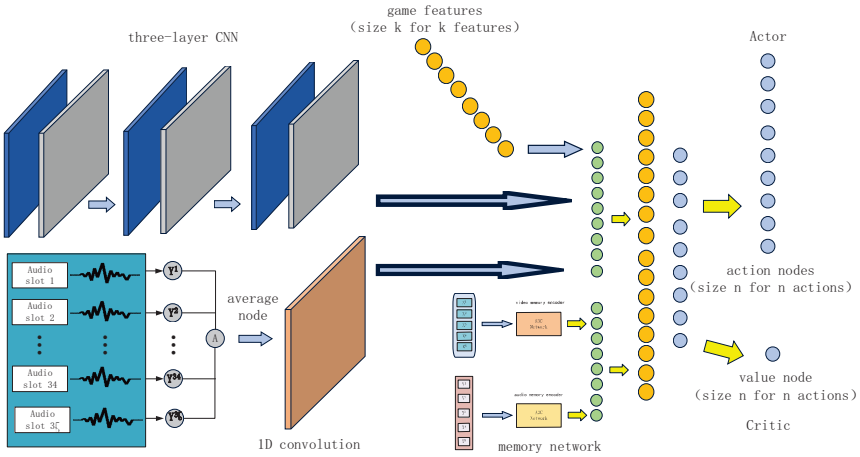

Figure 2 Memory-design network architecture.

Table 4 Units for magnetic properties

Features	Description
Kill count	Counts the number of monsters killed during the current episode. Killing other players/bots do not count towards this.
HITCOUNT -	Counts number of hit monsters/players/bots during the current episode
HEALTH	A player's health rate(0-100)
ARMOR	Armor left int player's current weapon
POSITION_X, POSITION_Y, POSITION_Z	Player's coordinates in 3D game space
ANGLE, PITCH, ROLL	Player's posture in 3D game space
VELOCITY_X, VELOCITY_Y, VELOCITY_Z	Player's velocity in 3D game space

full connected layer and is connected to a dense layer which represent the probability of action sets.

As Lample mentioned in his paper, game information could give a huge lift for a DRQN to converge. Although a lot of game information was available under the Vizdoom environment, considering designing a real combat agent just as in real battlefield, features that cannot be obtained in real combat scenario are abandoned; the game features we utilized are shown in Table 4.

To fully self-explore the agent, the tactics utilized in combat scenarios and sent to the agent are primarily the fundamental actions. The flexible use of these basic actions allows the agent to perform high quality tactical actions,

Table 5 Action design

Action	Parameters	Description
Move	Directions	Forward, Backward, Right, Left
Additional Move	Type	Jump, Crouch, Turn 180 degree[2]
Posture	Directions	Look Up, Look_Down
Zoom	Velocity	Speed Rate
Weapon	Type	Attack, Reload, Strafe

Table 6 Reward design

Type	Parameters	Description
Shoot	Number	The bullets number agent shoot
Be shot	Number	The bullets number agent is shot
Positive advantage	Bool	Situation that agent can see its enemies, enemies can't see it
Ammo amount less than 10	Bool	Ammo amount is less than 10 or not
Ammo amount more than 10	Bool	Ammo amount is more than 10 or not

such as run and shoot, hide and shoot. Simultaneously, the action search space is reduced by limiting the action types. The design of the action space is shown in Table 5.

In a deathmatching scenario, a combat agent's ultimate goal is to protect itself and kill more enemies. Therefore, designing the reward function is vital during the agent's training. Traditional reward functions typically reward the winner of the final game. However, such a strategy has very sparse rewards, making it difficult for the algorithm to converge. To overcome this problem, we define the agent's actions during a deathmatching game that led to final positive or negative rewards, such as taking an advantage situation, changing barrages in time and enhancing shooting precision by implementing additional moves. The rewards of the corresponding posture are generated by evaluating the posture int the deathmatching game.

Table 6 shows the reward function designed to fully use the knowledge of death-matching game.

5 Experiment Evaluation

In this section, the proposed architecture is evaluated in two environments: Vizdoom and a high precision simulation platform (HPSP). The empirical results are presented in Sections 5.1 and 5.2 respectively. The overall view of our experiment is shown in Figure 3.

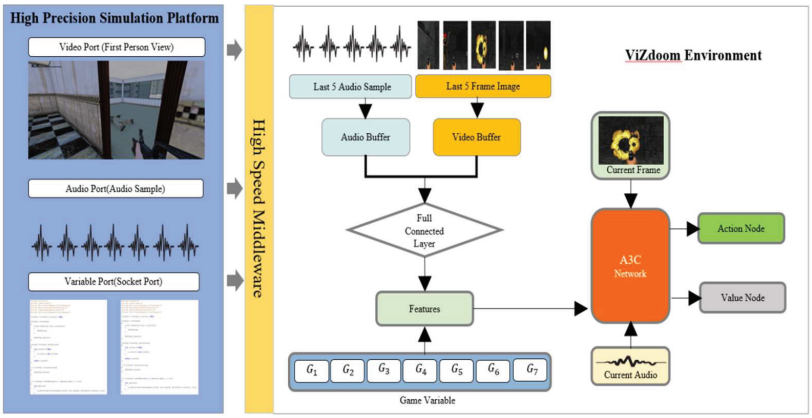

Figure 3 Experiment overview.

5.1 Vizdoom Environment

A server computer equipped with a NVIDIA GeForce RTX 3080 Ti Laptop GPU with 32 GB memory is used for both training and testing. Python 3.7 and Pytorch are used for programming on Ubuntu OS.

First, our memory-decision network is compared with models without memory-decision module, then compared proposed model with built-in models, such as the direct future prediction model proposed by Chaplot. Our code can be found at Github.

We first train a model without memory and audio modules, which means the input of the network is only pixel raw data, we choose RMSPropOptimizer as our optimizer with learning rate to 0.00001, which was selected based on grid search experiments over a range of values (0.0001, 0.00001, 0.000001) to balance convergence speed and stability. This choice aligns with prior work on A3C-based frameworks, where small learning rates are preferred for stable policy updates in asynchronous training. The network model architecture is shown in Figure 4.

The training results are shown below.

Proposed model is then trained with memory and audio modules, which means the input of the network is images and audios. The network model architecture was shown before in Figure 1. The frames number is changed to find out the impact of the memory buffer size; 3 frames and 5 frames are used as parameters to tune and the results are shown in Figures 6 and 7.

As shown in Figures 8–10, the performance of our model with a memory module is compared with a model using only a current image as an input

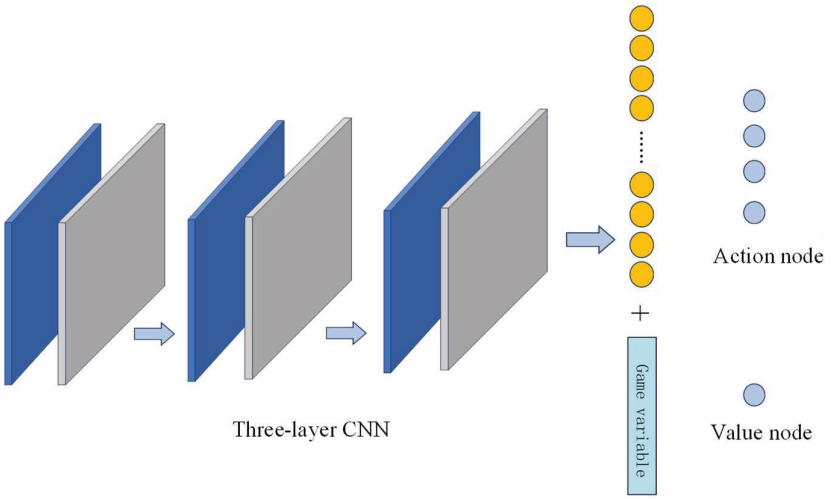

Figure 4 Network without memory input.

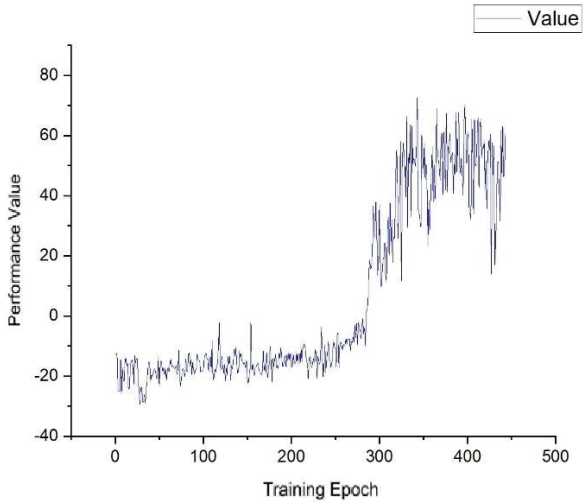

Figure 5 Performance value of the network model without memory.

of the model. It is very clear that our model could reach performance of the same level in a much shorter time, and its performance level is almost the same as the model without a memory module. This demonstrates that the proposed model has a huge advantage over the model without a memory module in convergence time. This advantage will significantly accelerate the

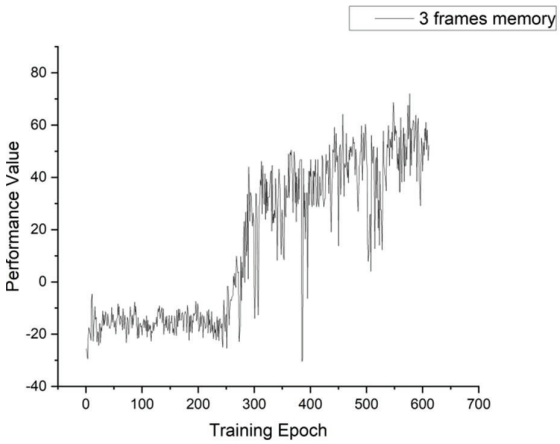

Figure 6 Performance of model with 3 frames memory.

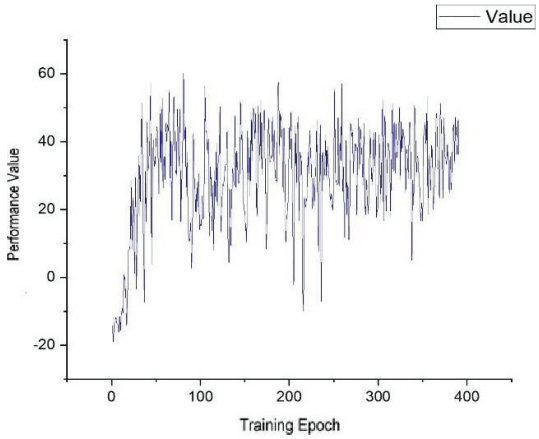

Figure 7 Performance of model with 5 frames memory.

adapting speed not only of the simulation agents but also the real autonomous equipment in a battlefield.

5.2 High Precision Simulation Platform

To further verify the feasibility of our proposed method with higher confidence and strong adversariality, and to broaden the application scenarios of our agent model, parallel simulation experiments are conducted based on a high-precision simulation platform. Entity and situation information between

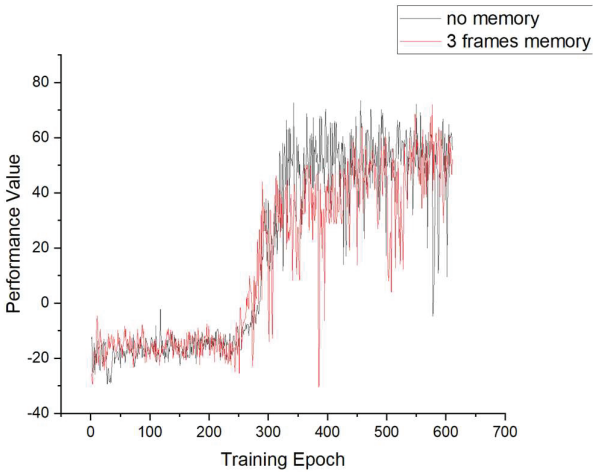

Figure 8 Comparison of 3 frames memory and zero memory.

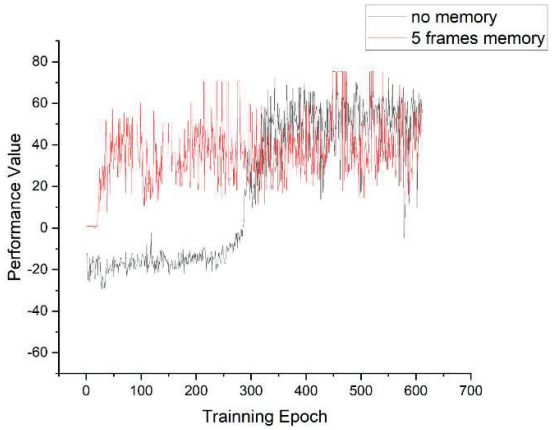

Figure 9 Comparison of 5 frames memory and zero memory.

both environments are transferred through high-speed interconnection middleware. The advantage of this approach lies in its ability to leverage both the inherent advantages of Vizdoom in providing good support for artificial intelligence algorithms and the advantages of our self-developed simulation platform, which boasts rich military rules and precise calculations from its high-precision physics engine.

A four-story building (Figures 11–13) is selected in an urban environment as our parallel environment for agent training, and utilized the platform's

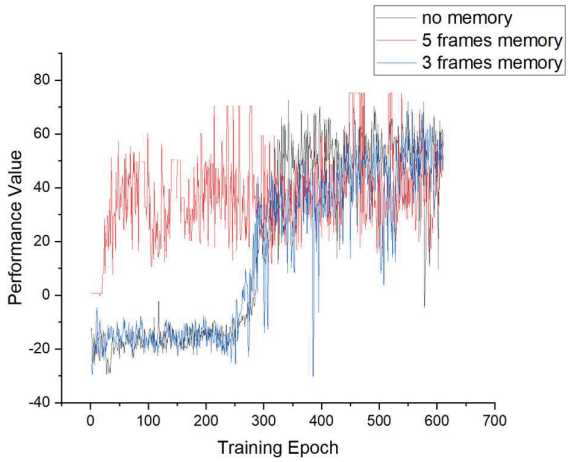

Figure 10 Overall comparison of 3 different models.

Figure 11 Building used to train the agent in a high precision simulation platform.

terrain editing tool to modify its internal structure to be consistent with that in Vizdoom, while also setting its materials and collision properties.

By placing a combat agent equipped with a rifle in the high precision simulation platform and synchronizing its position, ammunition, and health values using middleware, first-person video images, sound information, and battlefield environment data is transmitted from the platform to the Memory-Decision network model built in the Vizdoom platform according to our model. Unlike the Vizdoom platform, the opponents in the high-precision simulation platform are not randomly generated but are pre-deployed and driven by a finite state machine as a behavior model. In our scenario, the opponents are deployed in rooms inside buildings and perform actions such as alerting, striking, and evading under the drive of the state transition model.

Figure 12 Internal structure of the building.

Figure 13 Internal structure of the room.

Figure 14 Agent engaging with enemies.

We first engaged in confrontation with opponent agents (Figure 14) driven by a finite state machine in the high-precision simulation platform and collected data. Subsequently, agents are integrated into our Memory-Decision network model for further confrontation. 500 trials are conducted under the guidance of both models, and average results are shown in Figures 15 and 16.

From the simulation results, the performance of proposed agent model in carrying out building clearance tasks is significantly better than that of the

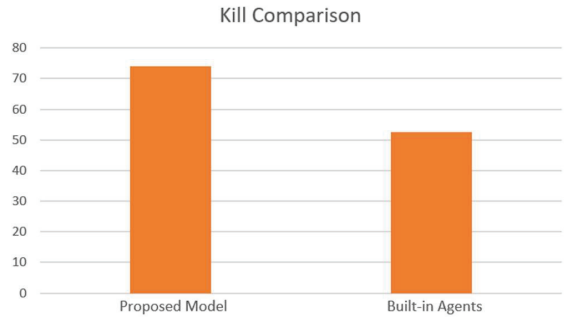

Figure 15 Kill comparison of the proposed model and built-in agents.

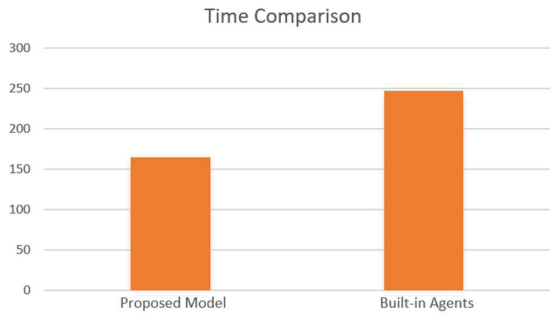

Figure 16 Time comparison of the proposed model and built-in agents.

built-in finite state machine based agent. The task execution time is shortened by 33%, and the killing efficiency is improved by 24.1%. This proves that our model not only performs well in the Vizdoom environment, but also demonstrates good results in practical applications on our high-precision simulation platform, which can significantly shorten the development time of simulation applications and improve the performance level of intelligent agents.

6 Conclusion

In this work, an agent model designed for combat simulation in a 3D virtual environment based on a network architecture with a memory decision-making module is proposed. Validation is conducted in the Vizdoom environment, and the results demonstrate that our proposed model exhibits significant advantages in training speed and convergence time within the Vizdoom environment. Furthermore, this model is embedded into our high-precision

simulation platform using high-speed interconnection middleware, enabling the transmission and reception of data such as first-person perspective images of agents, sound waveform data, and battlefield situations through ports. We conducted confrontation experiments in a multi-agent environment featuring high-precision physical structures and materials, and driven by finite state machines, and compared our proposed agent architecture with the original agent driven by finite state machines. The results also indicate that our proposed agent architecture demonstrates significant advantages in multiple dimensions, including lethality, survivability, and ammunition consumption.

Our model can to some extent improve the weak intelligence and long construction time of current combat simulation intelligent agent behavior modeling, and effectively enhance the realism of intelligent agents in the adversarial process. This is of great significance for enhancing the intelligence and combat level of opponents in combat evaluation simulation and simulation training applications. In the future, based on this research, we will combine the generalization ability of big prediction model technology to combat tasks and environmental factors, and expand the research object to not only individual intelligent agents, but also multiple types of intelligent agents such as drones and unmanned vehicles, to carry out further research and experiments.

7 Additional Information

Tianci Zhang and Hao Fang conceived the idea. Yongyong Wei built a simulation environment and complete the experiment. Tianci Zhang designed the memory-decision network and its training algorithm. Hao Fang and Yongyong Wei integrated the model into Vizdoom and HPSP virtual environment. Tianci Zhang and Hao Fang wrote the paper with support from Yongyong Wei.

Tianci Zhang and Hao Fang and Yongyong Wei contributed equally to this work.

References

[1] P. K. Davis, 'Military Applications of Simulation: A Selected Review', Applied System Simulation: Methodologies and Applications, 2003: 407–435.

[2] M. McPartland, M. Gallagher, 'Reinforcement learning in first person shooter games', IEEE Transactions on Computational Intelligence and AI in Games, 2010, 3(1): 43–56.
[3] V. Mnih, K. Kavukcuoglu, D. Silver, et al., 'Playing atari with deep reinforcement learning', arXiv preprint arXiv:1312.5602, 2013.
[4] C. Berner, G. Brockman, B. Chan, et al., 'Dota 2 with large scale deep reinforcement learning', arXiv preprint arXiv:1912.06680, 2019.
[5] J. Dansie, 'Game Development in Unity: Game Production, Game Mechanics and the Effects of Gaming', 2013.
[6] O. Vinyals, I. Babuschkin, W. M. Czarnecki, et al., 'Grandmaster level in StarCraft II using multi-agent reinforcement learning', nature, 2019, 575(7782): 350–354.
[7] R. S. Sutton, 'Reinforcement learning, a Bradford book', 1998.
[8] N. Osman, C. Sierra, 'Autonomous agents and multi-agent systems', Kluwer Academic Publishers, Massachusetts, 2008.
[9] A. Tolk, 'Engineering Principles of Combat Modeling and Distributed Simulation', 2012.
[10] J. Schoffel, 'Half-life 2: deathmatch. Australian PC User, 2005.
[11] R. Chin, 'First-Person Shooter Game Framework', Beginning iOS 3D Unreal Games Development, 2012: 283–318.
[12] E. Larsen, 'Unreal tournament iii', Personal Computer World, 2008, 31(5), p. 77.
[13] D. Long, M. Fox, 'Progress in AI planning research and applications', UPGRADE: The European Journal for the Informatics Professional, 2002, 3(5): 10–25.
[14] M. Humphrys, 'Action selection methods using reinforcement learning', From Animals to Animats, 1996, 4: 135–144.
[15] T. Bewley, J. Lawry, 'Richards A. Modelling agent policies with interpretable imitation learning', International Workshop on the Foundations of Trustworthy AI Integrating Learning, Optimization and Reasoning. Cham: Springer International Publishing, 2020: 180–186.
[16] H. M. Le, Y. Yue, P. Carr, et al., 'Coordinated multi-agent imitation learning', International Conference on Machine Learning. PMLR, 2017: 1995–2003.
[17] M. L. Littman, 'Markov games as a framework for multi-agent reinforcement learning', Machine learning proceedings 1994. Morgan Kaufmann, 1994: 157–163.

[18] M, Hannebauer, J, Wendler, E, Pagello, et al., 'Situation based strategic positioning for coordinating a team of homogeneous agents' //Balancing Reactivity and Social Deliberation in Multi-Agent Systems: From RoboCup to Real-World Applications. Springer Berlin Heidelberg, 2001: 175–197.

[19] L. Zhang, Y. Gu, X. Zhao, et al., 'Generalizing soft actor-critic algorithms to discrete action spaces', Chinese Conference on Pattern Recognition and Computer Vision. Singapore: Springer Nature Singapore, 2024: 34–49.

[20] P. R. Gorton, A. Strand, K. Brathen, 'A survey of air combat behavior modeling using machine learning', arXiv preprint arXiv:2404.13954, 2024.

[21] V. Karpov, V. Vorobiev, et al., 'About some aspects of finite state machine models application to group control'. Mekhatronika, Avtomatizatsiya, Upravlenie, 2023.

[22] J. Li, J. Su, Q. Gu, et al., 'Behavior Tree Generation Study for Multi-agent', International Conference on Man-Machine-Environment System Engineering. Singapore: Springer Nature Singapore, 2024: 503–509.

[23] S. Taghipour, H. A. Namoura, M. Sharifi, et al., 'Real-time production scheduling using a deep reinforcement learning-based multi-agent approach', INFOR: Information Systems and Operational Research, 2024, 62(2): 186–210.

[24] S. N. A. Jawaddi, A. Ismail, 'ntegrating OpenAI Gym and CloudSim Plus: A simulation environment for DRL Agent training in energy-driven cloud scaling', Simulation Modelling Practice and Theory, 2024, 130: 102858.

[25] T. Papagiannis, G. Alexandridis, A. Stafylopatis, 'Boosting Deep Reinforcement Learning Agents with Generative Data Augmentation', Applied Sciences, 2023, 14(1): 330.

[26] D. Ye, G. Chen, W. Zhang, et al., 'Towards playing full moba games with deep reinforcement learning', Advances in Neural Information Processing Systems, 2020, 33: 621–632.

[27] S. Risi, M. Preuss, 'From chess and atari to starcraft and beyond: How game AI is driving the world of AI', KI-Künstliche Intelligenz, 2020, 34(1): 7–17.

[28] C. Badica, L. Braubach, A. Paschke, 'Rule-based distributed and agent systems', Rule-Based Reasoning, Programming, and Applications: 5th International Symposium, RuleML 2011–Europe, Barcelona, Spain,

July 19-21, 2011. Proceedings 5. Springer Berlin Heidelberg, 2011: 3–28.

[29] E. Bonabeau, 'Agent-based modeling: Methods and techniques for simulating human systems', Proceedings of the national academy of sciences, 2002, 99(suppl_3): 7280–7287.

[30] C. M. Macal, M. J. North, 'Tutorial on agent-based modeling and simulation', Proceedings of the Winter Simulation Conference, 2005. IEEE, 2005: 14 pp.

[31] Bohemia Interactive. 'Virtual Battle Space 4 product brochure', 2024. Retrieved from https://://bisimulations.com/products/vbs4.

[32] M. Kempka, M. Wydmuch, G. Runc, et al., 'Vizdoom: A doom-based ai research platform for visual reinforcement learning', 2016 IEEE conference on computational intelligence and games. IEEE, 2016: 1–8.

[33] G. Lample, D. S. Chaplot, 'Playing FPS games with deep reinforcement learning', Proceedings of the AAAI conference on artificial intelligence. 2017, 31(1).

[34] Unity, 'Unity ML-Agents Introduction', 2024. Available: https://unity.cn/product/machine-learning-agents.

[35] H. da Silva Corrêa Pinto, L. O. Alvares, 'An extended behavior network for a game agent: An investigation of action selection quality and agent performance in unreal tournament', Mexican International Conference on Artificial Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005: 287–296.

[36] K. Adil, F. Jiang, S. Liu, et al., 'Training an agent for fps doom game using visual reinforcement learning and Vizdoom', International Journal of Advanced Computer Science and Applications, 2017, 8(12).

[37] S. Hegde, A. Kanervisto, A. Petrenko, 'Agents that listen: High-throughput reinforcement learning with multiple sensory systems',2021 IEEE Conference on Games. IEEE, 2021: 1–5.

[38] Poznan University of Technology. Vizdoom Documentation. 2024. Retrieved from https://vizdoom.cs.put.edu.pl/.

Biographies

Tianci Zhang was born in Haerbin, China in 1991. He received a B.Sc. degree in automation from Hangzhou Dianzi University, Hangzhou, China in 2013, an M.Sc. degree in artificial intelligence from Beijing Institution of Technology, Beijing in 2015, and is currently pursuing a Ph.D. degree in artificial intelligence from Beijing Institution of Technology. He has authored more than 10 articles. His research interests include agent modeling, optimization algorithms, and military simulation.

Yongyong Wei was born in Guangshui, China in 1978. He received a B.Sc. degree in electronic information from the Beijing Institution of Technology in 2009. He has been a researcher in agent modeling and simulation application designing at Ordnance Science and Research Academy of China since 2010. He has authored more than 10 articles. His research interests include multi-domain simulation, LVC simulation, and complex system evaluation.

Hao Fang was born in March 1973, and is a postdoctoral fellow, professor, and doctoral supervisor. He obtained a doctoral degree from Xi'an Jiaotong University in 2002 and worked as a postdoctoral fellow at INRIA Sophia Antipolis in France from April 2002 to July 2003. From November 2003 to December 2004, he worked as a postdoctoral fellow in the LASMEA laboratory at the CNRS National Research Center in France. He started teaching at Beijing Institute of Technology in 2005. His main research directions are intelligent robots, parallel robots, and multi-agent systems.