Training an Agent for FPS Doom Game using Visual Reinforcement Learning and VizDoom

Article *in* International Journal of Advanced Computer Science and Applications · December 2017

DOI: 10.14569/IJACSA.2017.081205

Adil Khan
City University of Science and Information Tec…
11 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Shaohui Liu
Harbin Institute of Technology
92 PUBLICATIONS **881** CITATIONS
SEE PROFILE

Seungmin Rho
Sungkyul University
62 PUBLICATIONS **210** CITATIONS
SEE PROFILE

Training an Agent for FPS Doom Game using Visual Reinforcement Learning and VizDoom

Khan Adil, Feng Jiang,
Shaohui Liu*, Aleksei Grigorev
School of Computer Science
and Technology
Harbin Institute of Technology
NO. 92, Xidazhi Street, Harbin,
Heilongjiang, China

B.B. Gupta
National Institute of
Technology
Kurukshetra,
India

Seungmin Rho
Sungkyul University,
Media Software, 306 Sungkyul Hall
Sungkyuldaehakro 53 Anyang,
Korea, 432749

***Abstract*—Because of the recent success and advancements in deep mind technologies, it is now used to train agents using deep learning for first-person shooter games that are often outperforming human players by means of only screen raw pixels to create their decisions. A visual Doom AI Competition is organized each year on two different tracks: limited death-match on a known map and a full death-match on an unknown map for evaluating AI agents, because computer games are the best test-beds for testing and evaluating different AI techniques and approaches. The competition is ranked based on the number of frags each agent achieves. In this paper, training a competitive agent for playing Doom's (FPS Game) basic scenario(s) in a semi-realistic 3D world 'VizDoom' using the combination of convolutional Deep learning and Q-learning by considering only the screen raw pixels in order to exhibit agent's usefulness in Doom is proposed. Experimental results show that the trained agent outperforms average human player and inbuilt game agents in basic scenario(s) where only move left, right and shoot actions are allowed.**

Keywords*—*Visual reinforcement learning; Deep Q-learning; FPS; CNN; computational intelligence; Game-AI; VizDoom; agent; bot; DOOM

I. Introduction

Doom is an FPS (First person shooter) game developed by Id-software. Its first installment was released on December 10, 1993, for the platform of 'DOS' and its second installment 'Doom II: Hell on Earth' was released in the following year (1994) for Microsoft Windows, play-station, and Xbox-360. Its third installment Doom-3 was released for Microsoft Windows on August 3, 2004, which was later adapted for Linux and MacOSX. Also, later on, 'Vicarious Visions' ported the game to the Xbox and released it on April 3, 2005. Now the very recent and latest installment is 'DOOM' developed by id-software and published by 'Bethesda Softworks'. It was released worldwide on Microsoft Windows, play-station 4 and X box-one as well on May 13, 2016. A common screen of the Doom game is shown in Fig. 1.

These days the research community is very active in research on 'Doom' for being a hot area of research using techniques like deep reinforcement learning or visual reinforcement learning. Besides, different Doom-based research platforms like 'VizDoom', 'CocoDoom' and 'ResearchDoom' [1] is developed for implementing deep learning techniques or methods. In the same way, every year different visual Doom AI competitions are organized where the agents (bots) are confirmed to exhibit human-like actions and to show that visual reinforcement learning in 3D FPS game environments is feasible.

Fig. 1. A typical Doom Game screen.

Like other domains, the deep learning has become well-known in computer video games as well, in showing improved performance than conventional approaches in managing high dimensional records such as bulky visual inputs [2]. Also, playing games in artificial intelligence (AI) has often been used as methods for benchmarking agents [3]-[6]. So, because of such reasons it was thought to propose deep learning with Q-learning in training the agent using the Doom-based research platform 'VizDoom', similar to the approach proposed in [7] but unlike in learning method (parameters) and the environment used for the experiments because the proposed agent's learning parameters, experimental environment, total learning and testing lasting time, and partial settings are different (see Section III) which is a part of the contribution to this paper. Further, in comparison to the total reward achieved by the authors agent, the proposed agent's total score is higher and always positive in numbers, also, initially the training reward of the author's agent is negative in numbers where the proposed agent training percentage is always positive in numbers. Besides, the proposed neural network architecture is also different than the one proposed by the authors. In order to introduce and explain the proposed work further in detail, this paper is organized into different sections. Section II presents related work, Section III describes the proposed method and experimental work and Section IV shows the results. Finally, Section V concludes the paper with the future work.

II. Related Work

The initial implementations of reinforcement learning based on visual inputs were performed in [8] and [9] in which the robots football-playing skills were developed. But since the availability of VizDoom for research community there are many other contemporary works on training a 'Doom AI' based agents using the VizDoom platform that includes the efforts in [10] where the authors presented CLYDE: a deep reinforcement learning Doom playing agent, participated in Visual Doom AI Competition held at the IEEE Conference on Computational Intelligence and Games 2016 where CLYDE competed with 8 other agents and managed to achieve 3rd place. Table I shows the CLYDE performance and results of the Visual Doom AI Competition 2016. Considering its relative simplicity and the fact that the authors deliberately avoided a high level of customization to keep the algorithm generic, it performed very well in a partially observable multi-agent 3D environment using Deep reinforcement learning techniques that have already been traditionally applied before in fully observable 2D environments. The CLYDE architecture is shown in Fig. 2 for further observations.

Similar to CLYDE, another agent called Arnold: a comprehensive and an independent agent for playing FPS games by means of screen raw pixels that exhibited the usefulness of Doom is presented in [11]. Arnold was trained using deep reinforcement learning by means of an 'Action-Navigation' structure that practices a distinct deep neural network for discovering the map and confronting the adversaries. The agent also utilized systems such as amplifying high level game characteristics, reward shaping and progressive updates for effective training and real performance where later Arnold outperformed typical human players and inbuilt game agents on different variation of death-match by obtaining the premier kill-to-death ratio in both tracks of the visual Doom AI Competition and was declared 2nd according to the number of frags. Table II shows the Arnold performance and results of the Visual Doom AI Competition 2016.

TABLE I. CLYDE Performance in Terms of a Total Number of Frags in Comparison with Other Bots in the Visual Doom AI Competition 2016

Place	**Bot**	**Total Frags**
1	F1	559
2	Arnold	413
3	CLYDE	393
4	TUHO	312
5	5Vision	142
6	ColbyMules	131
7	Abyssll	118
8	WallDestroyerXxx	-130
9	Ivomi	-578

TABLE II. Performance of Arnold On Both Tracks in Comparison with Other Bots in the Visual Doom-AI Competition [11]

	Limited DeathMatch		**Full Deathmatch**	
Agent Name	**No. of Frags**	**K/D Ratio**	**No. of Frags**	**K/D Ratio**
5Vision	142	0.41	12	0.20
AbyssII	118	0.40	-	-
Arnold	413	**2.45**	164	**33.40**
CLYDE	393	0.94	-	-
ColbyMules	131	0.43	18	0.20
F1	**559**	1.45	-	-
IntelAct	-	-	**256**	3.58
Ivomi	-578	0.18	-2	0.09
TUHO	312	0.91	51	0.95
WallDestroyerXxx	-130	0.04	-9	0.01

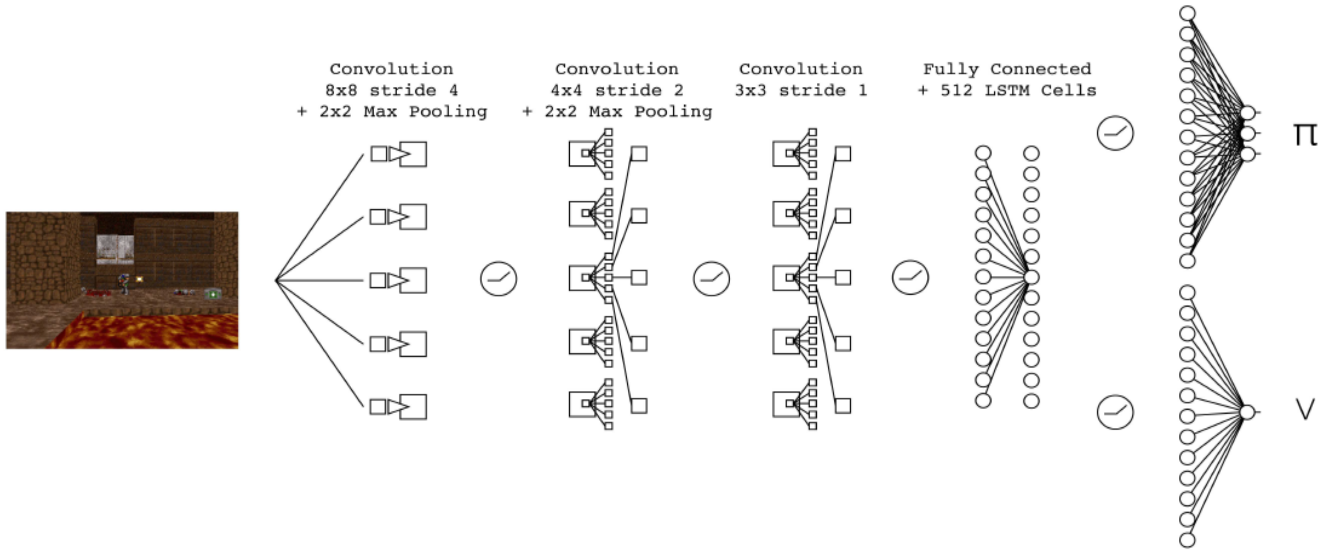

Fig. 2. An architecture of the agent CLYDE

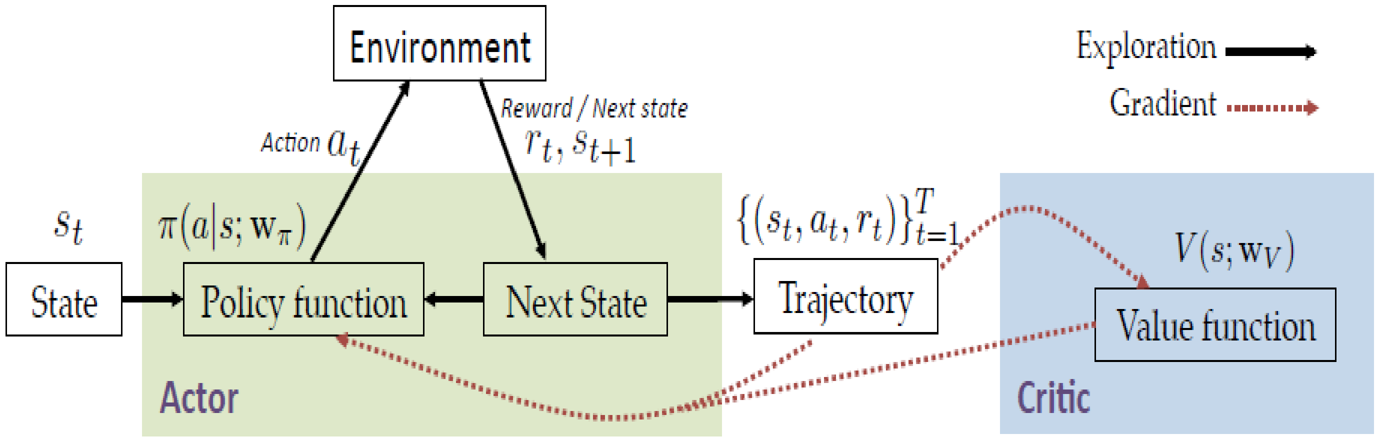

Fig. 3. The basic framework of Actor-critic model [13].

Similarly, a more related work to the proposed approach is performed in [12] where agents are trained for two different scenarios: a simple basic and a complex navigation maze problem using convolutional deep neural networks with Q-learning and experience replay. Where the trained agents were able to exhibit human-like behaviors.

A framework is proposed in [13] for training vision-based agents using deep learning and curriculum learning for FPS games that won the Track 1 by 35% higher score than the second place holding agent in 'VizDoom' AI competition 2016 on a known map. The framework combines the state-of-the-art reinforcement learning approach A3C model with curriculum learning. The model is simpler in design and uses game stats from AI only rather than using opponents' information [14]. The basic framework of the Actor-critic model is shown in Fig. 3 for understanding and further observations.

A. *Deep Reinforcement Learning*

The commonly applied techniques for learning agents or bots are deep reinforcement learning techniques that are logical and efficient in decision making. A similar deep reinforcement learning technique is employed in [15] for learning agents that can make generic and interactive decision making and whose mathematical framework is based on Markov Decision Processes (MDPs). An MDP is a tuple of different fields like (S, A, P, R, γ) where 'S' is the set of different states, 'A' is the set of different actions the agent can make at each time step t, 'P' is the transitional probability of moving from one state (s) to another state ($\acute{s}$) making an action (a), 'R' is the reward function representing that signal which the agent receives after doing different actions and changing states, and 'γ' is the discount factor. As usual, the goal of the reinforcement learning is to learn a policy π: s$\rightarrow$a that maximizes the overall expected discounted average reward over the agent run. A commonly used technique to learn such a policy is to learn the 'action value function' Q^{π}(s, a) iteratively. So as to gradually approximate the expected reward in a model-free fashion. The employed augmented framework is shown below in Fig. 4 that consistently learns better.

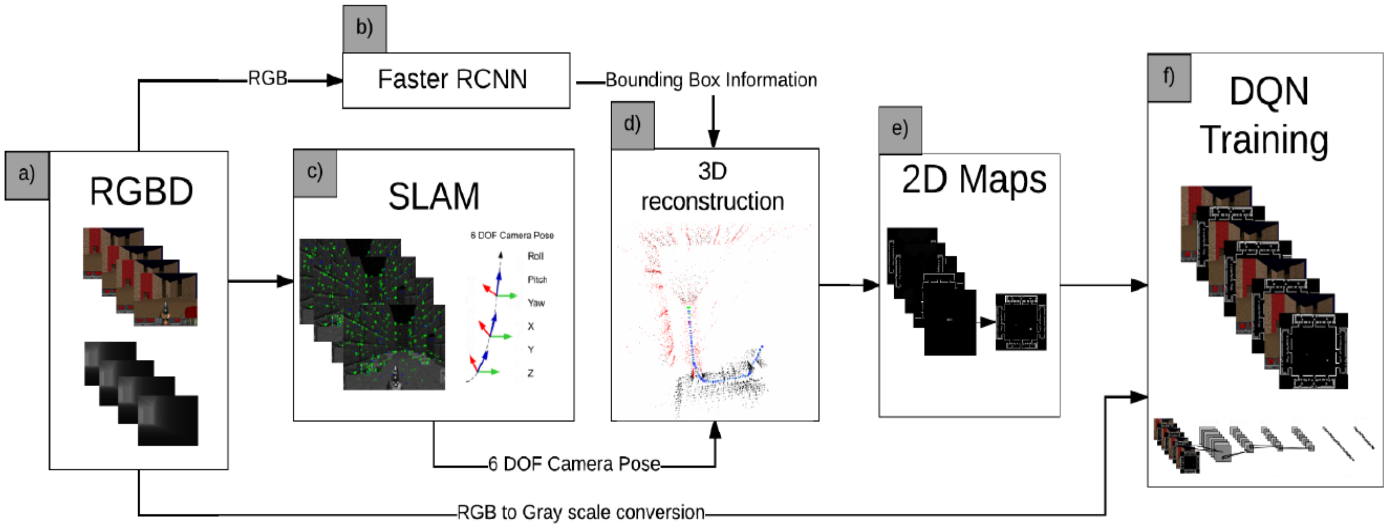

Fig. 4. Framework overview (a) Observing image and depth from VizDoom, Running Faster-RCNN (b) for object detection and SLAM (c) for pose estimation. Doing the 3D reconstruction (d) using the pose and bounding boxes. Semantic maps are built (e) from projection and the DQN is trained (f) using these new inputs [15].

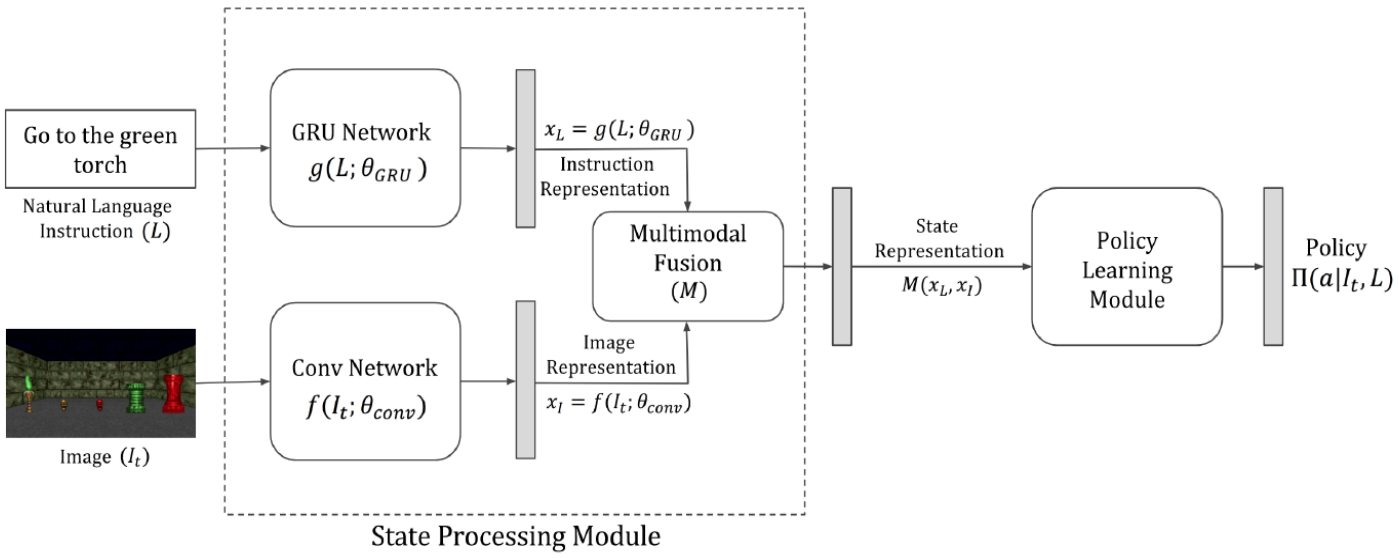
Go to the green torch
Natural Language Instruction (L)
Image (I_t)
GRU Network $g(L;\theta_{GRU})$
Conv Network $f(I_t;\theta_{conv})$
$x_L = g(L;\theta_{GRU})$
Instruction Representation
Multimodal Fusion (M)
Image Representation
$x_I = f(I_t;\theta_{conv})$
State Representation $M(x_L, x_I)$
Policy Learning Module
Policy $\Pi(a|I_t, L)$
State Processing Module

Fig. 5. The proposed model architecture [16] to estimate the policy given the natural language instruction and the image showing the first-person view of the environment.

A similar work (training an agent using VizDoom) of an end-to-end trainable neural structure for task-oriented language grounding in a 3D environment is proposed in [16] that supposes no prior linguistic or perceptual data and needs only raw pixels from the environment and the natural language instruction as input. The model combines the image and text representations using a Gated-Attention mechanism and learns a policy to implement the natural language instruction using standard reinforcement and imitation learning methods. The authors showed the usefulness of the suggested model on unseen instructions as well as unseen maps, both quantitatively and qualitatively. They also introduced a unique environment based on a 3D game engine to simulate the challenges of task-oriented language grounding over a rich set of instructions and environment states. The proposed model is shown in Fig. 5 for further details.

B. *Deep Q-Networks*

A model is trained in [14] to simultaneously learn game features statistics such as the existence of enemies or items along with minimizing Q-learning objective that showed a dramatic improvement in training speed and performance of the model. The authors proposed architecture is modularized to permit several models to be independently trained for different phases of the game. The architecture substantially outperformed built-in AI agents of the game and human players as well in death-match scenarios which is shown below in Fig. 6.

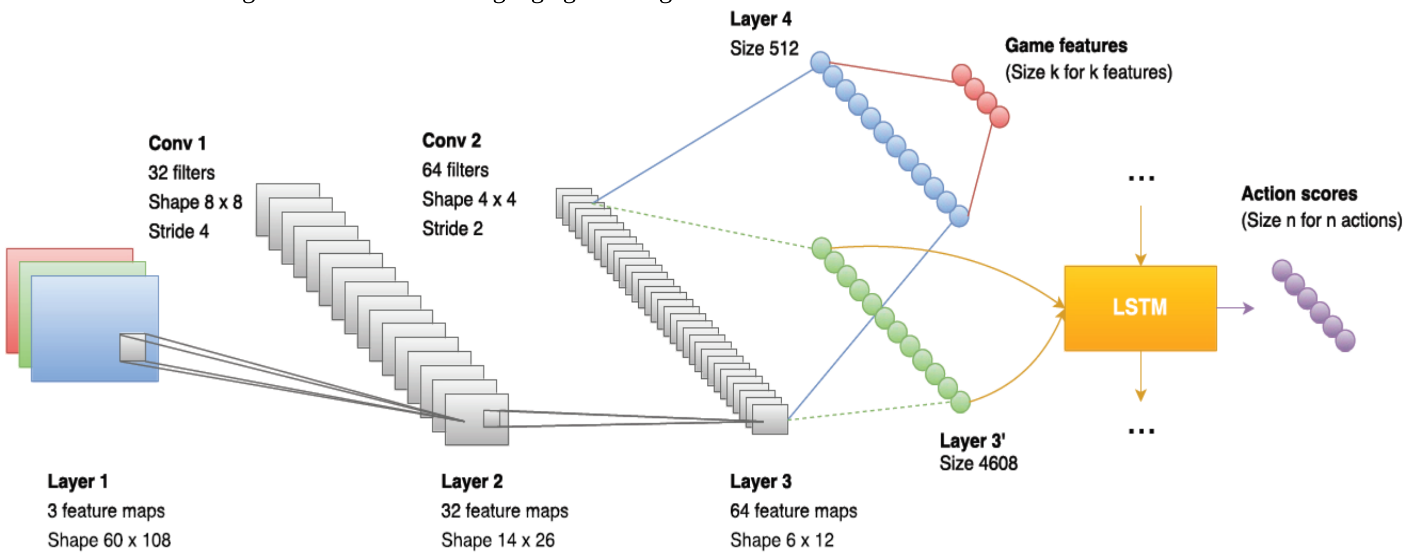

Fig. 6. The proposed architecture of the model in [14]. The Convolutional layers are given an input image. The output is split into two streams produced by the convolutional layers. The first one (bottom) flattens the output (layers 3) and inputs it to LSTM, as in the DQRN model. The second one at the top directs it to an extra hidden layer "layer 4", after then to a final layer representing each game features. While training, the game features and the Q-learning objectives are trained mutually.

A short summary of the DQN model followed in [14] is briefly presented here for supporting the proposed concept, where, learning a policy for an agent that maximizes the expected sum of discounted rewards $\mathbf{R_t}$ is dealt with Reinforcement learning which is represented mathematically as,

$$\mathbf{R}_t = \sum_{i=t}^{T} \gamma\acute{t} - t_{\gamma\acute{t}} \tag{1}$$

Where 'T' is the game termination time and $\gamma \in [0,1]$ is a discount factor that calculates the importance of future rewards. The Q-function for the expected return from performing an action 'a' in a state 's' for a given policy $\boldsymbol{\pi}$ is defined as:

$$\boldsymbol{Q^{\pi}(s,a)} = \mathbb{E}[\boldsymbol{R_t} | \boldsymbol{s_t}{=}\mathrm{s}, \boldsymbol{a_t} = \boldsymbol{a}] \tag{2}$$

The highest return can be expected to achieve by using an approximation function to estimate the activation-value function Q. Particularly neural network parametrized by $\boldsymbol{\theta}$ is used by DQN and the idea to obtain an estimate of the Q-function of the current policy that is close to the optimal Q-function $\boldsymbol{Q}^*$ by following a strategy [14],

$$\boldsymbol{Q}^*(\boldsymbol{s,a}) = \max_{\pi} \boldsymbol{E}[\boldsymbol{R_t} | \boldsymbol{s_t} = \boldsymbol{s}, \boldsymbol{a_t} = \boldsymbol{a}\,] = \max_{\pi} \boldsymbol{Q^{\pi}}(\mathrm{s, a}) \tag{3}$$

It can also be described as the goal to find $\boldsymbol{\theta}$ such that $\mathbf{Q}_{\boldsymbol{\theta}}$ (s, a) $\approx \mathbf{Q}^*(\mathbf{s,a})$. The optimal Q-function verifies the Bellman optimality equation

$$\mathbf{Q}^*(\mathrm{s, a}) {=} \mathbf{E}[\mathrm{r} {+} \boldsymbol{\gamma} \max_{\acute{\mathbf{a}}} \mathbf{Q}^*(\acute{\mathbf{s}}, \acute{\mathbf{a}}) \mid \mathrm{s, a}] \tag{4}$$

If $\mathbf{Q}_{\boldsymbol{\theta}} \approx \mathbf{Q}^*$, it is obvious to consider that $\mathbf{Q}_{\boldsymbol{\theta}}$ needs to be close in verifying it for Bellman equation that leads to the below loss function:

$$\mathbf{L_t}\,(\boldsymbol{\theta_t}) {=} \mathbb{E}_{\mathbf{s,a,r,\acute{s}}}[(\mathbf{y_t} {-} \mathbf{Q}_{\boldsymbol{\theta}_t}(\mathbf{s,a}))^2] \tag{5}$$

Here 't' is the current time step, and $\mathrm{y_t} = \mathrm{r} + \boldsymbol{\gamma} + \mathbf{max}_{\acute{a}}\, \boldsymbol{Q}_{\boldsymbol{\theta}_t}(\acute{\boldsymbol{s}}, \acute{\boldsymbol{a}})$. The value of $\mathrm{y_t}$ is fixed that leads to the following gradient.

$$\boldsymbol{\nabla_{\theta_t} L_t(\theta_t)} = \mathbb{E}_{\boldsymbol{s,a,r,\acute{s}}}[\boldsymbol{y_t} {-} \boldsymbol{Q_{\theta}(s,a)} \boldsymbol{\nabla_{\theta_t} Q_{\theta_t}}(\mathrm{s\,, a})\,] \tag{6}$$

The above gradient can also be computed using this below approximation.

$$\boldsymbol{\nabla_{\theta_t} L_t(\theta_t)} \approx (\boldsymbol{y_t} - \boldsymbol{Q_{\theta}(s,a)}) \boldsymbol{\nabla_{\theta_t} Q_{\theta_t}(s,a)} \tag{7}$$

Using Experience replay is a well-known concept for breaking the correlation between successive samples. An agent experiences $(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{r_t}, \boldsymbol{s_{t+1}})$ at each time step, are saved in the replay memory, where then the Q-learning updates are performed on batches of experiences arbitrarily sampled from the memory.

The $\boldsymbol{\epsilon} -$ **greedy strategy** is used to generate the next action at each training step with a probability $\boldsymbol{\epsilon}$ for selecting the next action randomly and with a probability $1 - \boldsymbol{\epsilon}$ according to the best action of the network. Practically it is common to start with $\boldsymbol{\epsilon} = \mathbf{1}$ which is decayed gradually [17].

C. Supervised Learning Techniques

A similar approach to training an agent via VizDoom platform is presented in [18] but using supervised learning techniques for sensorimotor control in immersive settings. The approach uses a high dimensional sensory stream and a lower-dimensional measurement stream. The cotemporal structure of the streams offers a rich supervisory signal that allows training a sensorimotor control model by communicating with the environment. The model learns to perform based on raw sensory input from a composite three-dimensional environment. The authors offered a formulation that permits learning without a fixed objective at training time and follows dynamic varying goals at a testing time. They also conducted a number of experiments in three-dimensional simulations based on classical FPS game Doom, the consequences demonstrated that the applied approach outperformed sophisticated earlier formulations specifically on exciting g and challenging tasks. The results also showed that trained models effectively generalize across environments and goals. The model trained with this approach won the full Death-match track of the Visual Doom AI Competition that was held in earlier unseen environments. The network structure the authors used in their experiments is shown below in Fig. 7.

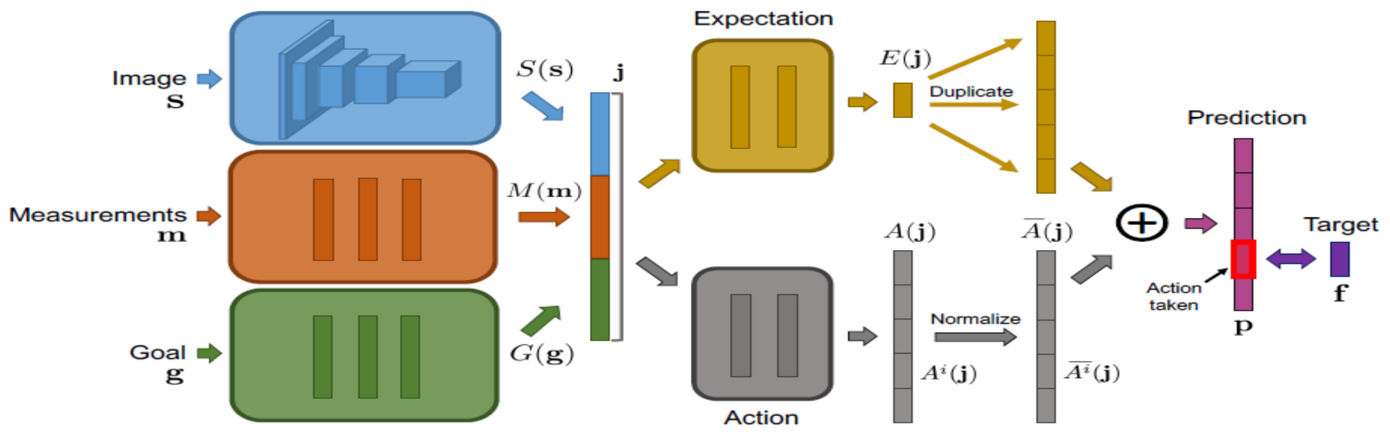

Fig. 7. Network Structure: the initial three input modules first process image 's', measurements m, and goal 'g' separately, then a joint representation j contains the concatenated output of these modules. Two parallel streams process the joint representation which predicts the normalized action-conditional differences { $\overline{A^i}\,(j)$ } and the expected measurements E(j) which are then joined to produce the concluding expectation for each action [18].

III. Proposed Method and Experimental Work

A. Basic Objective

The primary purpose of the experiments is to train a competent agent using visual reinforcement learning and 'VizDoom' for first-person shooter games, particularly 'Doom' to exhibit human-like behaviors and to outperform average human players and existing in-built game agents.

B. Scenario(s)

A rectangular chamber is used as a basic scenario (see Fig. 8) wherein the center of the room's long wall an agent spawns. Along the opposite wall, an immobile monster spawns at arbitrary positions. The agent moves towards the left side, right side and shoots as well. A single shot is sufficient to eradicate the monster. The episode finishes once the 300 frames are completed or the monster is either killed, whichever approach first. For killing the monster, the agent achieves 101 points, -5 for missing the shot and -1 for each individual action. But the best practice for the agent to learn killing the monster is to kill as rapidly as possible preferably with a solitary shot.

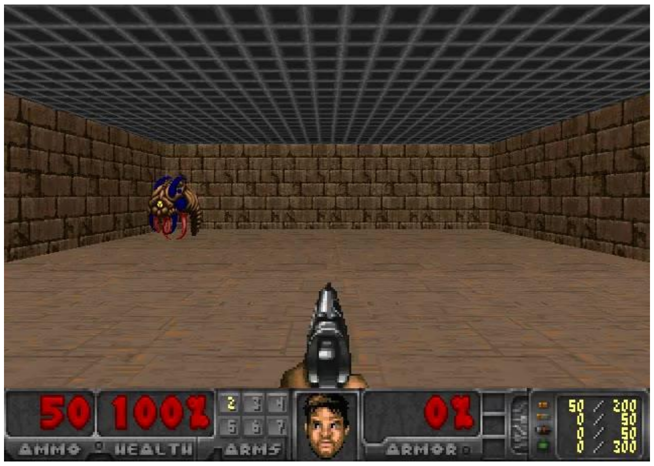

Fig. 8. The basic scenario used.

C. Deep Q-Learning

'Markov Decision Process' is used to model the problem and Q-learning to learn the policy. An ϵ-greedy policy with linear ϵ decay is used for selecting an action. The Q-function is approximated with the convolutional neural network by training it with 'Stochastic Gradient Decent' using experience replay.

D. Experimental Setup

- Neural Network Architecture

The network used in the experiments includes two convolutional layers with 32 square filters, 7 and 4 pixels wide, respectively, as shown in Fig 9. A max-pooling layer follows each convolutional layer with a max pooling of size 2 and Relu (Rectified Linear Unit) function for activation. Moreover, the network contains a fully connected layer with 800 leaky rectified linear units and an output layer with 8 linear units conforming to the 8 combinations of the 3 offered actions i.e. right, left and shoot.

- Learning Settings

In the experiments the discount factor is set to γ=0.99, learning rate α=0.00025, replay memory capacity of 10 000 elements, the resolution (30, 45), and mini-batch size to 64. The agent learned from 23, 239 steps consisting of performing an action, observing a transition, and updating the network. For monitoring the learning process, after each epoch (Approx. 2000 learning steps) 100 testing episodes are played.

- Environment used for Experiment

The experiments are performed in 'Pycharm professional 2016.3 version using ViZDoom 1.1.1, OpenCV 3.3, CMake 2.8+, Make, GCC 4.6+, Boost libraries 1.54+, and Python 3.5 (64-bit) with Numpy on an Ubuntu 16.04.3 installed computer with Intel® Core™ i7-7700 CPU @3.60 GHz x 8 and NVIDIA GeForce GTX 1080/PCIe/SSE2 GPU for processing CNN's, the whole learning process along with the 100 testing episodes lasted for almost 30.70 Minutes.

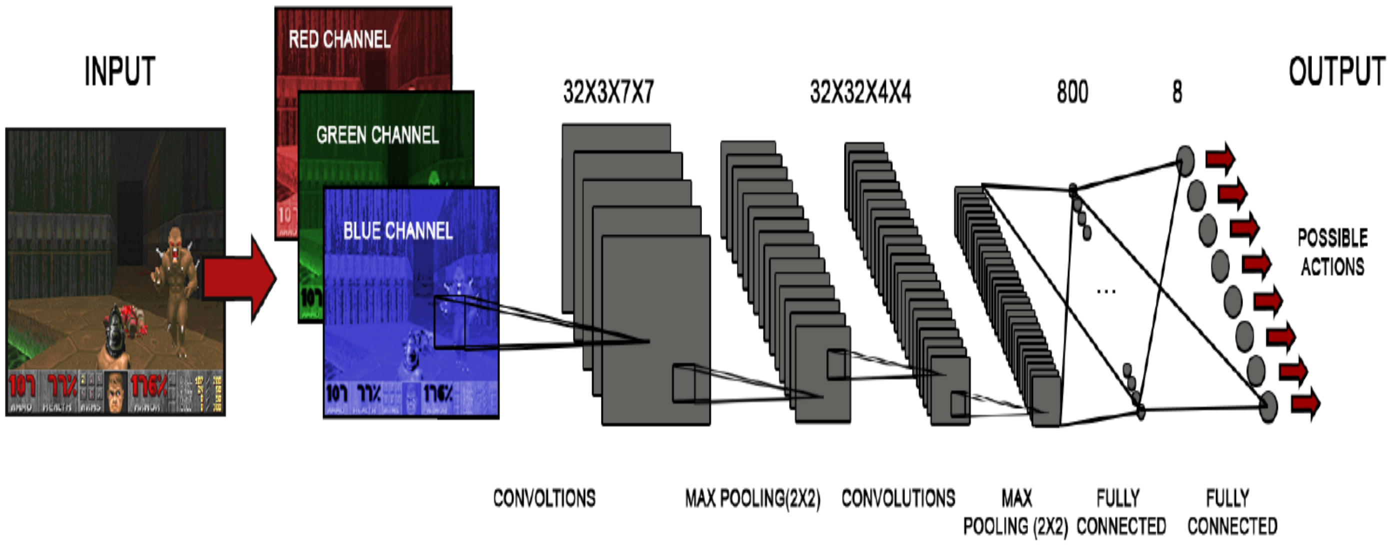

Fig. 9. CNN architecture used in the experiments.

IV. Results

In the experiments, a total of 10,188 training episodes (Basic scenarios) are played. The agent learning details are presented in Table III and explained as follows.

A. Learning

The total number of epochs performed are 20 shown in Column “Ep #”. The number of steps learned by the agent is obvious in column “SL/2,000” wherein the initial epochs the learning remains low, but improved in the following epochs, although in some epochs the learning remains unsuccessful. Similarly, the learning percentage also improved progressively and reached almost 100 %, which could be well understood and observed in Fig. 10. Further, column “IPS” represents the iterations per second performed in each learning Epoch. A different number of episodes (basic scenarios) are played during each epoch. The minimum, maximum and mean values are the actual learning and testing output achieved during each epoch by the agent and are displayed in their corresponding columns under learning and testing results in the table. The ‘ETT’ represents the agent elapsed testing time in minutes. The learning steps are kept limited to 2,000 for each epoch in the current experiments which will be kept large and dynamic for different scenarios like rocket basic, deadly corridor, defend the center, defend the line, and health gathering in the future work in order to train and develop competitive agents.

TABLE III. Agent Learning Results **EP#**: Epoch Number, **SL/2,000**: Steps Learned Out of 2,000, **LP%**: Learning in Percentage, **IPS**: Iterations Per Second, **EET**: Elapsed Testing Time (Minutes) **UL**: Unsuccessful Learning

Ep #	**SL/ 2,000**	**LP (%)**	**IPS**	**EP**	**Learning Results (performance)**			**Testing Results (performance)**			**ETT**
					Mean	**Min**	**Max**	**Mean**	**Min**	**Max**	
1				140	-132.6±187.3	-380.0	95.0	-161.1±232.4	-410.0	95.0	1.04
2	1285	64	45.28	133	-149.9±186.3	-375.0	95.0	-36.8±183.8	-410.0	95.0	2.56
3	1995	100	42.30	149	-111.9±177.1	-385.0	95	17.6±134.5	-360.0	95.0	3.47
4	1989	99	41.69	153	-106.3±170.6	-385.0	95.0	72.1±21.6	-18.0	95.0	4.35
5	1999	100	40.17	195	-48.8±141.6	-380.0	95.0	78.5±13.4	18.0	95.0	5.25
6	UL	-	UL	283	7.1±97.6	-390.0	95.0	72.6±17.5	-12.0	95.0	6.21
7	1996	100	41.58	352	29.4±77.1	-365.0	95.0	75.7±15.7	-18.0	95.0	7.21
8	1993	100	38.83	397	38.7±62.6	-355.0	95.0	73.3±17.7	-9.0	95.0	8.24
9	1998	100	36.33	433	45.1±47.6	-216.0	95.0	75.4±15.0	42.0	95.0	9.28
10	1997	100	38.00	544	58.7±32.7	-151.0	95.0	77.2±14.8	42.0	95.0	10.39
11	1997	100	31.93	600	63.7±27.9	-115	95.0	77.3±12.6	45.0	95.0	11.53
12	UL	-	UL	678	69.5±23.1	-129.0	95.0	77.1±12.1	45.0	95.0	12.72
13	UL	-	UL	757	73.9±16.4	-6.0	95.0	77.3±12.4	45.0	95.0	13.97
14	1999	100	30.87	754	73.8±17.4	-16.0	95.0	76.2±12.8	25.0	95.0	15.21
15	1993	100	32.90	749	73.4±17.3	-40.0	95.0	76.9±13.1	25.0	95.0	16.47
16	UL	-	UL	770	74.7±16.6	-61.0	95.0	77.7±11.1	45.0	95.0	17.71
17	UL	-	UL	773	75.0±15.1	-4.0	95.0	79.4±12.0	20.0	95.0	18.97
18	UL	-	UL	757	74.3±16.4	-25.0	95.0	78.5±10.7	59.0	95.0	20.22
19	UL	-	UL	774	75.2±14.2	-1.0	95.0	79.5±10.7	52.0	95.0	21.48
20	1998	100	28.35	797	76.2±14.9	-21.0	95.0	79.1±10.3	52.0	95.0	22.76

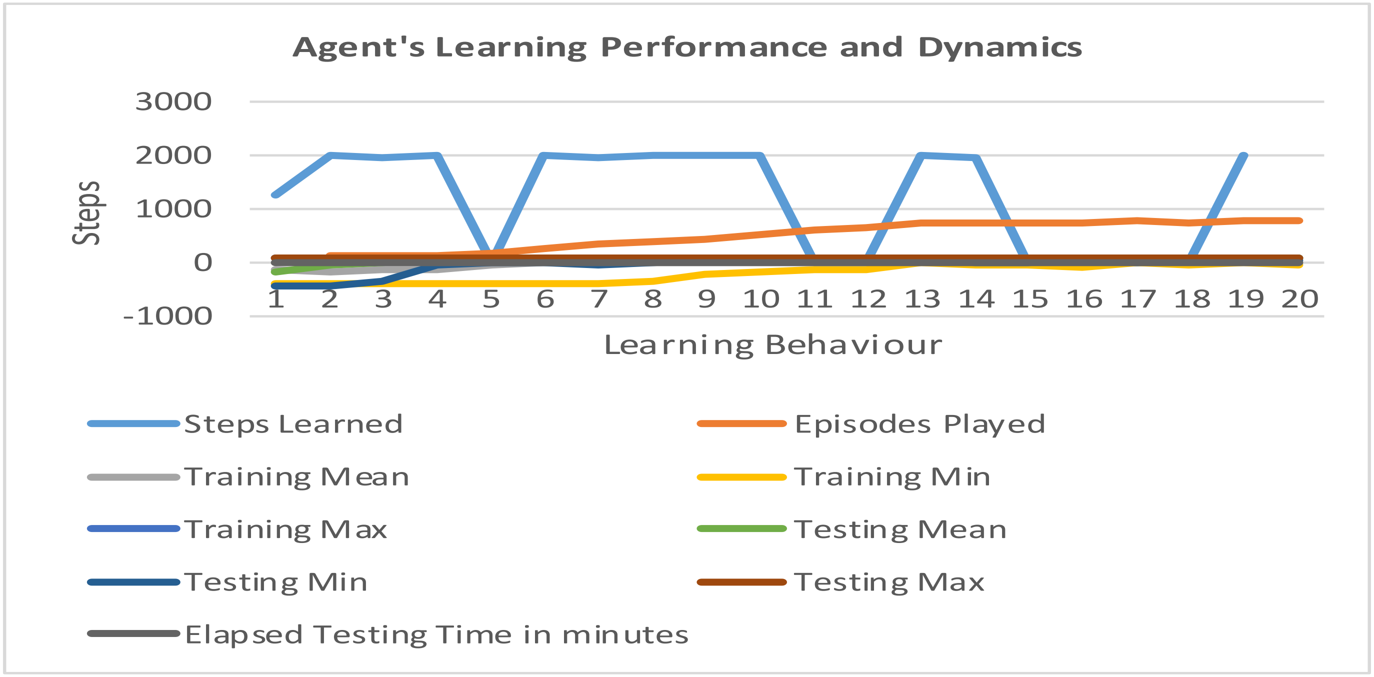

Fig. 10. Agent's learning performance and dynamics on basic scenario(s).

B. *Final Testing*

Similarly, in final testing phase, the agent is tested on 100 basic scenario(s) once the whole training finished (after 20th epoch), the agent's total score after each testing episode is shown in Table IV and its performance can be well understood and observe from the graph shown in Fig. 11.

As it is obvious in the graph that the agent behavior in shooting the spawning monster is balanced and its minimum, maximum and average shooting scores are 17, 94 and 74 which shows that the performance of the agent in basic 'move and shoot' scenario(s) is more decent and optimum because the agent is always tested even after each epoch while training in order for monitoring and observing its performance that is always found improved gradually with the passage of time. But as far as the agent overall testing output is concerned with basic scenario(s) so it performed well by moving to the proper position and shooting accurately.

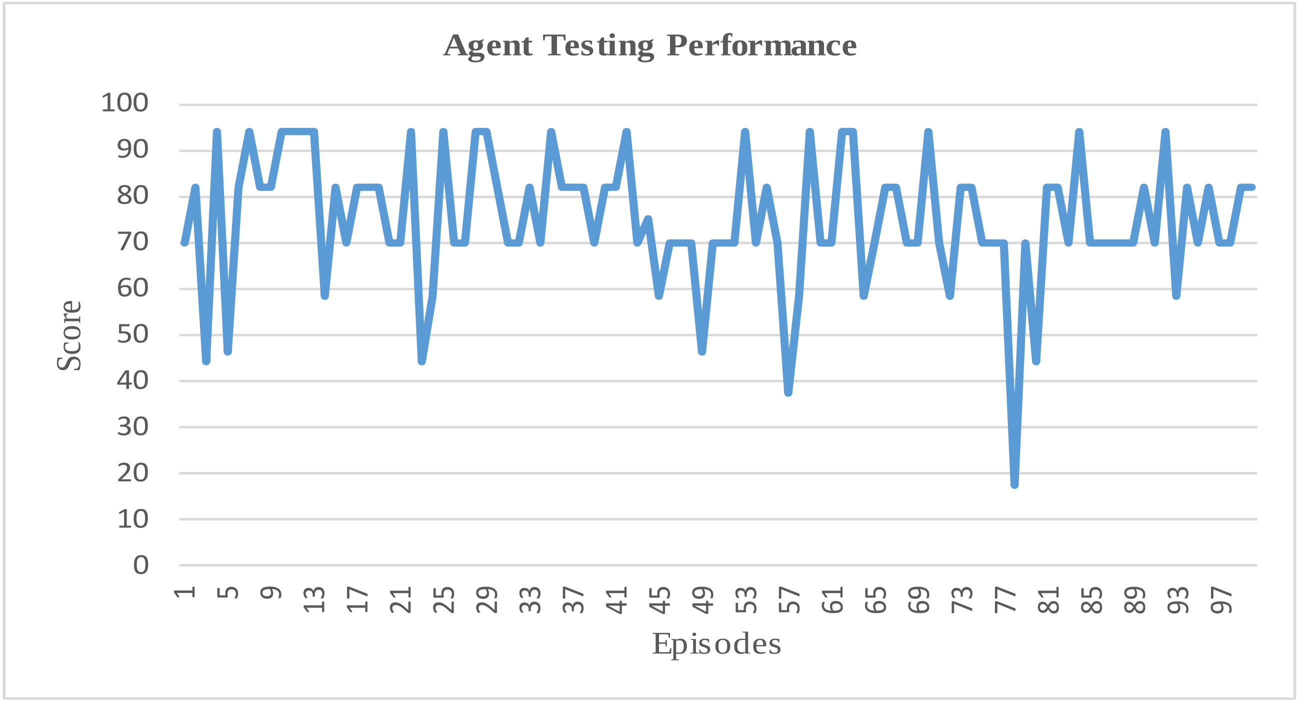

Fig. 11. Agent Testing Performance on 100 Basic Scenario(s).

TABLE IV. AGENT'S TESTING SCORES IN 100 EPISODES (SCENARIOS)

Episode No.	Total Score	Episode No.	Total Score	Episode No.	Total Score
1	70.0	35	94.0	69	70.0
2	82.0	36	82.0	70	94.0
3	44.0	37	82.0	71	70.0
4	94.0	38	82.0	72	58.0
5	46.0	39	70.0	73	82.0
6	82.0	40	82.0	74	82.0
7	94.0	41	82.0	75	70.0
8	82.0	42	94.0	76	70.0
9	82.0	43	70.0	77	70.0
10	94.0	44	75.0	78	17.0
11	94.0	45	58.0	79	70.0
12	94.0	46	70.0	80	44.0
13	94.0	47	70.0	81	82.0
14	58.0	48	70.0	82	82.0
15	82.0	49	46.0	83	70.0
16	70.0	50	70.0	84	94.0
17	82.0	51	70.0	85	70.0
18	82.0	52	70.0	86	70.0
19	82.0	53	94.0	87	70.0
20	70.0	54	70.0	88	70.0
21	70.0	55	82.0	89	70.0
22	94.0	56	70.0	90	82.0
23	44.0	57	37.0	91	70.0
24	58.0	58	58.0	92	94.0
25	94.0	59	94.0	93	58.0
26	70.0	60	70.0	94	82.0
27	70.0	61	70.0	95	70.0
28	94.0	62	94.0	96	82.0
29	94.0	63	94.0	97	70.0
30	82.0	64	58.0	98	70.0
31	70.0	65	70.0	99	82.0
32	70.0	66	82.0	100	82.0
33	82.0	67	82.0		
34	70.0	68	70.0		

V. CONCLUSION AND FUTURE WORK

In this paper, an agent is trained using Deep Q-Learning and 'VizDoom'. The agent is tested for almost 2000 (finally on 100 Ep) Doom scenarios where it demonstrated an intelligent behavior and the results achieved are better and positive in numbers than the results proposed by Hyunsoo and Kyung-J. K. 2016. After the scientific analysis, monitoring and observations of the simple 'move and shoot' basic scenario(s) results, it is also observed that the speed of the learning system largely rely on the quantity of frames the agent is permitted to skip while learning, in particular skipping frames from 4 to 10 are profitable, which is the future considered work, but this time with larger number of learning steps and Doom scenarios (episodes) by allowing the agent to access the sound buffer as presently agents are deaf.

ACKNOWLEDGMENT

The authors would like to thank NVIDIA for GPU donation. Also, this work is partially funded by the MOE–Microsoft Key Laboratory of Natural Language Processing and Speech, Harbin Institute of Technology, the Major State Basic Research Development Program of China (973 Program 2015CB351804) and the National Natural Science Foundation of China under Grant No. 61572155, 61672188 and 61272386.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] Mahendran, Aravindh, Hakan Bilen, João F. Henriques, and Andrea Vedaldi, ResearchDoom and CocoDoom: Learning Computer Vision with Games. arXiv preprint arXiv:1610.02431, 2016.

[2] Kulkarni, Tejas D., Ardavan Saeedi, Simanta Gautam, and Samuel J. Gershman, Deep successor reinforcement learning. arXiv preprint arXiv:1606.02396, 2016.

[3] Georgios N. Yannakakis, M., IEEE, and Julian Togelius, Member, IEEE, A Panorama of Artificial and Computational Intelligence in Games.pdf. IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, VOL. 7, NO. 4, 2015. 7 (4): p. 19.

[4] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S, Human-level control through deep reinforcement learning. Nature, 2015. 518 (7540): p. 529-533.

[5] Schaeffer, Jonathan, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen, Checkers is solved. science, 2007. 317 (5844): p. 1518-1522.

[6] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S, Mastering the game of Go with deep neural networks and tree search. Nature, 2016. 529 (7587): p. 484-489.

[7] Hyunsoo Park, a.K.-J.K., Deep Q-Learning using Redundant Outputs.pdf. IEEE Conference on Computational Intelligence and Games (CIG'16), 2016.

[8] Asada, Minoru, Eiji Uchibe, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. A vision-based reinforcement learning for coordination of soccer playing behaviors. in Proceedings of AAAI-94 Workshop on AI and A-life and Entertainment. 1994. Seattle, USA.

[9] Asada, Minoru, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda., Purposive behavior acquisition for a real robot by vision-based reinforcement learning. Machine learning, 1996. 23 (2): p. 279-303.

[10] Ratcliffe, Dino, Sam Devlin, Udo Kruschwitz, and Luca Citi, Clyde: A deep reinforcement learning DOOM playing agent. What's Next For AI In Games: AAAI 2017 Workshop. 2017.San Francisco, USA., 2017.

[11] Chaplot, D.S. and G. Lample. Arnold: An Autonomous Agent to Play FPS Games. in AAAI. 2017.

[12] Kempka, M., Wydmuch, M., Runc, G., Toczek, J. and Jaśkowski, W, ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. arXiv preprint arXiv:1605.02097, 2016.

[13] Wu, Y. and Y. Tian, Training agent for first-person shooter game with actor-critic curriculum learning. conference paper at ICLR 2017, 2017.

[14] Lample, G. and D.S. Chaplot, Playing FPS games with deep reinforcement learning. arXiv preprint arXiv:1609.05521, 2016.

[15] Bhatti, S., Desmaison, A., Miksik, O., Nardelli, N., Siddharth, N. and Torr, P.H, Playing Doom with SLAM-Augmented Deep Reinforcement Learning. arXiv preprint arXiv:1612.00380, 2016.

[16] Chaplot, D.S., Sathyendra, K.M., Pasumarthi, R.K., Rajagopal, D. and Salakhutdinov, R, Gated-Attention Architectures for Task-Oriented Language Grounding. arXiv preprint arXiv:1706.07230, 2017.

[17] Hafner, D., Deep Reinforcement Learning From Raw Pixels in Doom. arXiv preprint arXiv:1610.02164, 2016.

[18] Dosovitskiy, A. and V. Koltun, Learning to act by predicting the future. arXiv preprint arXiv:1611.01779, 2016.

AUTHORS' PROFILE

Adil Khan received B.Ed from the University of Peshawar, BS Honors in computer science from Edwards college Peshawar, M.S Degree in Computer Science from City university of science and information technology Peshawar, C.T. from AIOU Islamabad, MCSE from Microsoft and CCNA from Cisco. In 2014-2016, he was a Senior Lecturer in Higher Education Department KPK, Pakistan. Currently, he is a Ph.D. Research Scholar at the school of computer science and technology, Harbin Institute of Technology, Harbin 150001 PR China. He is interested in Artificial Intelligence, Neural networks, Real Time Strategy Games, First-Person-Shooter Games, machine learning, computer vision and image processing.

Feng Jiang received B.S., M.S., and Ph.D. Degrees in Computer Science from Harbin Institute of Technology (HIT), Harbin 150001 PR China, in 2001, 2003, and 2008, respectively. He is now an Associate Professor in the Department of computer science, HIT and a visiting scholar in the school of electrical engineering, Princeton University. He is interested in computer vision, pattern recognition, Game-AI, image and video processing.

Shaohui Liu received B.S., M.S., and Ph.D. Degrees in Computer Science from Harbin Institute of Technology (HIT), Harbin 150001 PR China, in 2000, 2002, and 2007, respectively. He is now an Associate Professor in the Department of computer science, HIT. He is interested in Game AI, data compression, pattern recognition, image and video processing.

Aleksei Grigorev received the Master Degree from Department of Mathematics and School of Computer Science and Technology, Harbin Institute of Technology (HIT), Harbin, China, in 2016. Currently, he is a Ph.D. Scholar at the school of computer science and technology, Harbin Institute of Technology, Harbin 150001 PR China. His research interests include Game AI, computer vision, pattern recognition, image and video processing.

Seungmin Rho received MS and Ph.D. Degrees in Computer Science from Ajou University, Korea, in 2003 and 2008, respectively. In 2008-2009, he was a Postdoctoral Research Fellow at the Computer Music Lab of the School of Computer Science at Carnegie Mellon University. In 2009-2011, he had been working as a Research Professor at the School of Electrical Engineering, Korea University. In 2012, he was an assistant professor of Division Information and Communication at Baekseok University. Currently, Dr. Rho is a faculty member of the Department of Multimedia at Sungkyul University. His research interests include Game AI, multimedia systems, machine learning, knowledge management as well as computational intelligence.