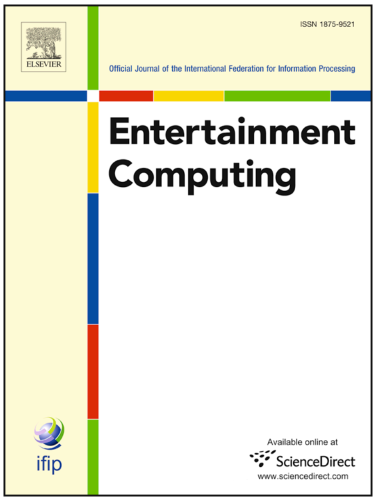

Playing First-Person Shooter Games with Machine Learning Techniques and Methods Using the VizDoom Game-AI Research platform

Adil Khan, Muhammad Naeem, Muhammad Zubair Asghar, Aziz Ud Din, Atif Khan

PII:	S1875-9521(19)30049-7
DOI:	https://doi.org/10.1016/j.entcom.2020.100357
Reference:	ENTCOM 100357
To appear in:	*Entertainment Computing*
Received Date:	2 April 2019
Revised Date:	14 January 2020
Accepted Date:	15 February 2020

Playing First-Person Shooter Games with Machine Learning Techniques and Methods Using the VizDoom Game-AI Research platform

Adil Khan[1,2], Muhammad Naeem[1], Muhammad Zubair Asghar[3], Aziz Ud Din[2], Atif Khan[4]

[1]Department of Computer Science, University of Peshawar, KP, Pakistan

[2]Department of Computer Science, SZIC, University of Peshawar, KP, Pakistan

[3]Institute of Computing and Information Technology, Gomal University, D. I. Khan, KP, Pakistan

[4]Department of Computer Science, Islamia College, Peshawar, KP, Pakistan

adil.adil25@yahoo.com / adil@uop.edu.pk / Dradil@hit.edu.pk

Abstract Artificial Intelligence in the form of machine learning is employed in games to control non-human computer-players, agents or bots. However, most of these games such as Atari took place in 2D environments that were fully observable to the agents. Currently, it is of extreme significance to employ such machine learning techniques and methods in 3D environments such as Doom. Therefore, In this paper, we train agents on the health gathering scenario of the classical first-person shooter game Doom by first presenting the Direct Future Prediction to train an agent that uses a simple architecture with no additional supervisory signals, then differentiate and compare the performance of the agents trained by using several different machine learning techniques, and the AI reinforcement learning platform 'VizDoom', a 3D partially observable environment, with interesting enhanced properties that makes agents to stand out from inbuilt AI agents and human players. We have continued to use computer games as a benchmark for the performance of AI as having been so successful in the past. We also compared the results of our findings to conclude the performance of the agents trained with different machine learning techniques. The agents performed well against both human players and inbuilt game agents.

Keywords Artificial Intelligence, Artificial Neural Network, Autonomous Systems, Computational Intelligence, Intelligent agents, Visual Deep Reinforcement Learning, Machine Learning

1. INTRODUCTION AND RESEARCH MOTIVATION

In the last few decades, due to the progress in artificial intelligence, a revolution and sudden change have been observed in the technology both in hardware and software [1]. This change is seeping and taking over in our lives up to a certain extent, affecting how we live, work and entertain ourselves such as employing domestic robots servants, healthcare uses, electronic trading, remote sensing, expert systems, traffic control systems, autonomously-powered self-driving vehicles, and from behavioral algorithms to suggestive searches [2], etc. In the same way, gaming is a widely recognized part of our cultural landscape and as old as our human ancestors. The earliest computers were very slow and the interaction with the user was limited to basic principles. In the early '40s, computers evolved, and programmers started to develop new virtual worlds and surprising ways of interaction between the user and the machine [3]. But now due to advancements in technology such as GPU's[4], TPU's[5] and the revolution in deep neural networks [6] it has become possible for artificial intelligence to step-in in video games as well where massive graphical data in the form of frames, or to be more specific a huge amount of multidimensional data is required to be processed and execute [7]. In the recent past, machine learning techniques and methods were employed in Atari games for training agents, where later, the agents performed on 49 different Atari games with better and improved results. However, most of these Atari games took place in 2D environments that were fully observable to the agents [8]. Currently, it is of extreme significance to employ such machine learning techniques and methods in 3D environments such as Doom [9] a first-person-shooter game shown in fig. 1, Starcraft [10] a third person shooter game based on real-time strategies, and sandbox open-world games such as Grand Theft Auto V and Minecraft [11] because the research community in AI think and consider that computer video games are the best test-beds for testing different artificial intelligence techniques, methods, and algorithms before evaluating them in real-world life. Thus, in this paper, state-of-the-art machine learning techniques that were before partially tested in 2D environments are now employed in a 3D environment known as Doom, to train, differentiate and compare agents performances, such as advantage actor-critic (A2C) [12], advantage actor-critic long short-term memory (A2C-LSTM) [13], asynchronous advantage actor-critic (A3C)

[14], Deep Q-network (DQN) [15], Deep recurrent Q-network (DRQN) [16], Double deep Q-network (DDQN) [17], C51-DDQN [18], Dueling deep Q-network (DDQN) [19], and Reinforce [20] whereafter applying most of the agents are found useful and effective. In addition, this paper presents one of the 4 best techniques that performed well on the VizDoom AI platform [21]. It was suggested that making such research available is beneficial for the community researching on first-person-shooter games which may set up a base for further research and improvement.

ACKNOWLEDGMENT
This work is funded by the University of Peshawar, Pakistan and the Higher Education Department KPK, Pakistan. The authors would also like to acknowledge and thank NVIDIA Corporation for two powerful sets of GPU machines donation.

AUTHORS PROFILES

Dr. Adil Khan

http://orcid.org/0000-0003-2862-5718 (ORCID ID)

He received C.T. from AIOU Islamabad, B. Ed from the University of Peshawar, BS Honors in Computer Science from Edwards College Peshawar, M.S in Computer Science from City University of Science and Information Technology Peshawar and Ph.D. from University of Peshawar, Peshawar, Pakistan. From 2014-2016, he was a Lecturer in Higher Education Department KPK, Pakistan and from 2016-2019 he was serving as a research scholar at the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001 PR China. Currently Adil khan is working as an Assistant Professor at the Department of Computer Science, SZIC, University of Peshawar. He has published many publications in top-tier academic journals and conferences and is interested in Machine Learning, Game Artificial Intelligence (Game-AI), Neural networks, Real Time Strategy Games, First-Person-Shooter Games, Sandbox open world Games, Computer Vision, Image Processing (Breast Cancer Detection). He can be reached at personal E-mail: adil.adil25@yahoo.com

Professor (Asst) & Dr. Muhammad Zubair Asghar

http://orcid.org/0000-0003-3320-2074 (ORCID ID)

He is an Assistant Professor at Institute of Computing and Information Technology, Gomal University, Dera Ismail Khan, KP, Pakistan and approved Ph.D. supervisor recognized by Higher Education Commission (HEC), Pakistan. His Ph.D. research interest includes Game-AI, First-person Shooter Games, Third-Person Shooter Games, Computational Intelligence, Computational Linguistics, Machine Learning, Text Mining, Opinion Mining, Sentiment Analysis, and Big Data Solutions for Social Networks. He has published more than 40 publications in journals of international reputation (JCR and ISI indexed). He has more than 15 years of University teaching and laboratory experience in Artificial Intelligence and Intelligent Systems Design. He is a Guest Editor of special issues in the journal of 'Social Computing in Health Informatics and Business Intelligence'. He is also a reviewer of many impact factor journals and an Associate Editor of IEEE ACCESS and Plos One. He can be reached at official E-mail: zubair@gu.edu.pk

Playing First-Person Shooter Games with Machine Learning Techniques and Methods Using the VizDoom Game-AI Research platform

Abstract Artificial Intelligence in the form of machine learning is employed in games to control non-human computer-players, agents or bots. However, most of these games such as Atari took place in 2D environments that were fully observable to the agents. Currently, it is of extreme significance to employ such machine learning techniques and methods in 3D environments such as Doom. Therefore, In this paper, we train agents on the health gathering scenario of the classical first-person-shooter game Doom by first presenting the Direct Future Prediction to train an agent that uses a simple architecture with no additional supervisory signals, then differentiate and compare the performance of the agents trained by using several different machine learning techniques, and the AI reinforcement learning platform 'VizDoom', a 3D partially observable environment, with interesting enhanced properties that makes agents to stand out from inbuilt AI agents and human players. We have continued to use computer games as a benchmark for the performance of AI as having been so successful in the past. We also compared the results of our findings to conclude the performance of the agents trained with different machine learning techniques. The agents performed well against both human players and inbuilt game agents.

Keywords Artificial Intelligence, Artificial Neural Network, Autonomous Systems, Computational Intelligence, Intelligent Agents, Machine Learning, Visual Deep Reinforcement Learning

1. Introduction and Research Motivation

In the last few decades, due to the progress in artificial intelligence, a revolution and sudden change has been observed in the technology both in hardware and software. This change is seeping and taking over in our lives up to a certain extent, affecting how we live, work and entertain ourselves such as employing domestic robots servants, healthcare uses, electronic trading, remote sensing, expert systems, traffic control systems, autonomously-powered self-driving vehicles, and from behavioural algorithms to suggestive searches, etc. In the same way, gaming is a widely recognized part of our cultural landscape and as old as our human ancestors. The earliest computers were very slow and the interaction with the user was limited to basic principles. In the early '1940s, computers evolved, and programmers commenced to develop new virtual worlds and surprising ways of interaction between the user and the machine. But now due to advancements in technology such as GPU's[1], TPU's[2] and the revolution in deep neural networks [3] it has become possible for artificial intelligence to step-in in video games as well where massive graphical data in the form of frames, or to be more specific a huge amount of multidimensional data is required to be processed and executed [4]. In the recent past, machine learning techniques and methods were employed in Atari games for training agents, where later, the agents performed on 49 different Atari games with better and improved results. However, most of these Atari games took place in 2D environments that were fully observable to the agents [5]. Currently, it is of extreme significance to employ such machine learning techniques and methods in 3D environments such as Doom [6] a first-person shooter game shown in fig. 1, StarCraft [7] a third-person shooter game based on real-time strategies, and sandbox open-world games such as Grand Theft Auto V [8] and Minecraft [9] because the research community in AI think and consider that computer video games are the best test-beds for testing different artificial intelligence techniques, methods, and algorithms before evaluating them in the real world. Thus, in this paper, state-of-the-art machine learning techniques that were before partially tested in 2D environments are now employed in a 3D environment known as Doom, to train, differentiate and compare agents performances, such as advantage actor-critic (A2C) [10], advantage actor-critic long short-term memory (A2C-LSTM) [11], asynchronous advantage actor-critic (A3C) [12], Deep Q-network (DQN) [13], Deep recurrent Q-network (DRQN) [14], Double deep Q-network (DDQN) [15], C51-DDQN [16], Dueling deep Q-network (DDQN) [17], and Reinforce [18] whereafter applying them most of the agents are found useful and effective. In addition, this paper presents one of the 4 best techniques that performed well on the VizDoom Game-AI research platform [19]. It was suggested that making such research available is beneficial for the community researching on first-person shooter games which may set up a base for further research and improvement.

1.1. The Academic Motivation of AI in Games

Why use machine learning (AI) techniques and algorithms to research on games? because the future belongs to artificial intelligence in games, particularly machine learning has immense potential and role in games designing and development. The possibilities abound, however, the challenges are also innumerable. Without a doubt, game development will experience a proliferation of these machine learning concepts, which is only a matter of time.

In addition, the more primary use of AI in games is to train games agents or bots in an intelligent way so that they could perform and act intelligently similar to human being's. By achieving such objectives, it creates more fun, challenge, and understanding to human players as for as playing or interacting with games is experienced and concerned. While playing against skilful human players AI agents or bots need to understand what a player does and how a player feels during the play. To gauge and enhance AI agent's vs human player's in-game experience, machine learning scientists, practitioners, games researchers, and developers use machine learning methods, such as reinforcement learning, supervised learning like support vector machines or neural networks to build and train the models to make them more effective and intelligent. Such advancements are particularly significant for the progress and development of computer video games industry.

Fig. 1. A sample screen from Doom showing the first-person perspective

2. Research On Doom Using the VizDoom Game-AI Research Platform

These days, game AI is one of the focused and active research areas in artificial intelligence as computer games are the best test-beds for testing theoretical ideas in AI before practically applying them in the real world. In this regard, many Game AI research platforms are familiarized for research on computer video games such as DeepMind, OpenAI gym, Unity and VizDoom which is based on first-person shooter (FPS) game Doom used for visual deep reinforcement learning from raw screen pixels in 3D game environments. The speed of the learning agent greatly depends on the number of frames the agent is permitted to skip. In other words, the frame skipping rate influences the agent's learning and final performance greatly particularly using deep Q-learning, experience replay memory, and the VizDoom Game AI research platform. The agents can be trained and tested on several of Doom's scenarios or maps in order to obtain good results and compare them with the existing state-of-the-art research work on Doom-based AI agents. So far the experiments performed on Doom's scenarios demonstrate that the profitable and optimal frame skipping rate falls in the range of 3 to 11 that provides the best balance between the learning speed and the final performance of the agent which exhibits human-like behaviour and outperforms an average human player and inbuilt game agents [20].

Moreover, there is a lot of existing research that relate to playing FPS games with visual deep reinforcement learning [21] such as the work introduced in [22], in which a method is presented to augment the models to exploit game feature information such as the presence of enemies or items. Similarly, another work described in [23] in which a competitive agent is proposed that is trained on the Doom's basic scenario(s) in the same semi-realistic 3D environment VizDoom using convolutional deep learning with Q-learning [24] that considers only the screens raw pixels for exhibiting agent's usefulness. The era of research using games changed when agents were trained using only the screen raw pixels.

Another more related work is proposed in [25] in which there is supposedly no reward signal (like there is in Atari games via the score). Instead, the authors used matching future state (measurements) predictions as a replacement to a reward signal. However, Probably, it should be noted that there is a constant health reduction in e.g. the basic room (walking on the radioactive ground) kind of resemblance to typical reinforcement learning r = -1 reward for each step when trying to reach a goal state quickly.

Sometimes Reinforcement Learning [26] environments with discrete actions are not getting it properly. They don't simulate human muscles easily. In other words, it's not easy for human players to wiggle the joystick at 114 microseconds between left and right as muscles get tired soon. Moreover, the author's proposed network doesn't employ LSTMs [27] to memorize transactions due to their proposed approach as the second-best agent in the visual Doom AI competition used LSTM but its simple feedforward architecture was 50% efficient.

In addition, each year a visual doom AI competition is organized for evaluating AI agents on two different tracks: limited death-match on a known map and a full death-match on an unknown map by using machine learning techniques. In this paper, agents are trained, tested, and compared using different machine learning techniques and methods. The methodology and experimental works are presented in several sections as follows.

2.1. Basic Objective

The purpose of the experiments is to train competent, well balanced and robust agents using machine learning techniques such as reinforcement learning and supervised learning [28] that can adapt to learn, act in complex and dynamic 3D environments. Such agents accept raw sensory input and core measurements to show that employed techniques are better at outperforming human players and other inbuilt game agents on the 'health gathering scenario(s)' of the VizDoom platform where only a few limited actions are allowed. Besides, the purpose also includes to prove, differentiate and compare several machine learning techniques by training agents on the VizDoom health gathering scenario(s).

2.2. VizDoom Health Gathering Environment (scenario)

We demonstrate implementing DFP on the health gathering scenario shown in fig. 2, provided by the VizDoom Game-AI research platform. The main objective of the agent is to survive as long as possible. However, at each time step, the agent's health decreases, so in order to live longer, the agent has to locate and pick up the health packs scattered across different parts of the environment or map. At the same time, the agent also needs to avoid running into poison jars which will take away its health. There are no active opponents in this simple health gathering scenario.

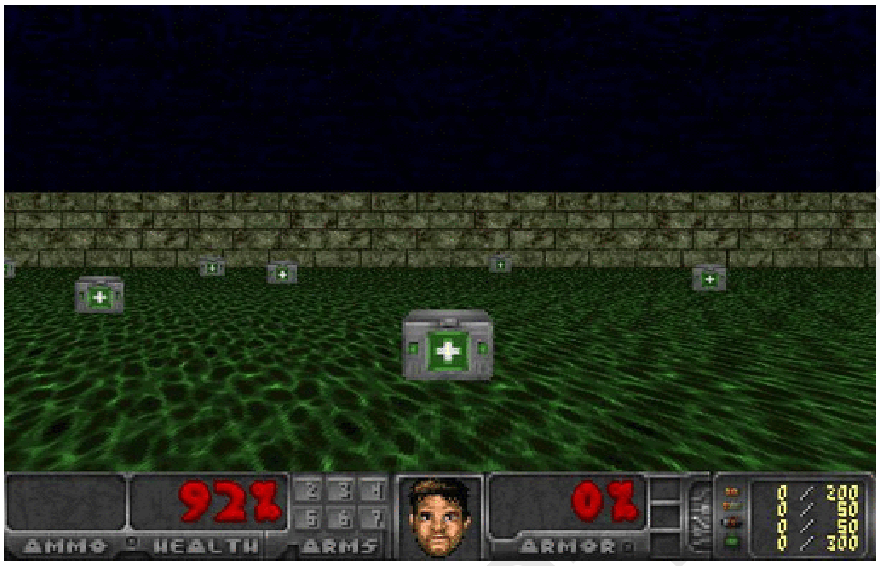

Fig. 2. The Health gathering scenario(s) used in the experiments

2.3. Machine Learning Approaches

Several Artificial Intelligence approaches are used to train agents such as using policy optimization [29] value optimization [30], and DFP as shown in fig. 3. In this paper, similar to traditional reinforcement learning, we suggest and compare training agents that learn through the response provided by interacting with the environment using machine learning techniques.

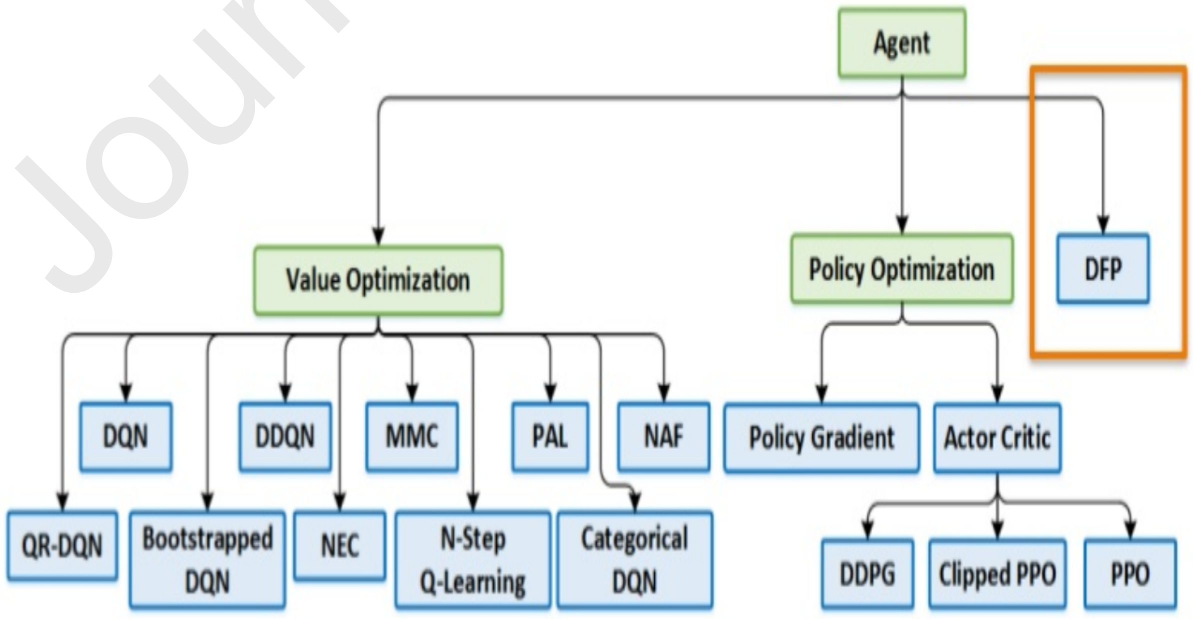

Fig. 3. Machine Learning Approaches

2.4. The Environment used for the Experiments

All the experiments are performed in Pycharm 2017.2 professional version using ViZDoom 1.1.5, Tensorflow 1.5.0 [31], Keras 1.2.2/2.0.5, OpenCV 3.3 [32], CMake 2.8+, GCC 4.6+, and Python 3.6 (64-bit) with Numpy on an Ubuntu Server 16.04.3 LTS Operating System with Intel® Core™ i7-7700 CPU @3. 60 GHz x 8 and NVIDIA GeForce GTX 1080/PCIe/SSE2 powerful GPU machine for processing the CNN's. The agents are trained for thousand to millions of steps consisting of performing actions, observing transitions, and updating the networks. The hyperparameter settings for all the experiments listed in Table 2 are met and found after hundreds of runs. The parameters are kept tuned until the models were found converged accurately and properly to meet the desired or expected results. The discount factor is set to γ=0.99 for almost all algorithms, the learning rate α varied from 0.001 to 0.0001, Experience replay buffer memory capacity is set from 50, 000 elements to 60,000, the screen-buffer is set to 640, 480 and remain the same for almost all of the algorithms, the batch-size is set to 32 and 64 for some algorithms, initial decay varied from 0.9 to 1, the final decay is set from 0.001 to 0.0001 and frame-per-action to 4 and 5. The overall learning and testing process is measured by the number of hours it takes to complete on a set of powerful GPU machines.

Table 1. Test Setup Hardware Specification

CPU	Intel® Core™ i7-7700 CPU @3.60 GHz x 8
GPU	NVIDIA GeForce GTX 1080/PCIe/SSE2 GPU
RAM	GiB DDR4

Parameters	**Algorithms/Methods**									
	DFP	**A3C**	**A2C**	**A2C-LSTM**	**DQN**	**DRQN**	**DuelingDQN**	**Double DQN**	**C51_ DDQN**	**Reinforce**
Discount Factor (γ)	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Learning Rate (α)	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.001	0.0001	0.0001
Experience	50,000	60,000	50,000	50,000	40,000	50,000	50,000	50,000	50,000	50,000

Replay Memory										
Screen Buffer	640, 480	640, 480	640, 480	640, 480	640, 480	640, 480	640, 480	640, 480	640, 480	640, 480
Batch Size	32	32	32	32	64	32	32	32	32	32
Initial Decay	1.0	0.9	1.0	0.9	1.0	1.0	1.0	1.0	1.0	1.0
Final Decay	0.0001	0.0001	0.0001	0.0001	0.001	0.0001	0.0001	0.0001	0.0001	0.0001
History length	4	4	4	4		6	4	4	4	4
Frames per Action	4	4	4	4	5	4	4	4	4	4
Time Elapsed (Hours)	12	12	16	20	170	12	12	12	22	19

Table. 2. Hyperparameters used in the experiments for training models (Agents or Bots)

3. Direct Future Prediction (DFP)

Direct future prediction (DFP) is a machine learning technique and it has one of the major ability and benefit to pursuing complex goals at test time. Normally, in reinforcement learning settings, learning is guided by a series of scalar reward signals [33], but in complex environments, the scalar rewards can be sparse and delayed, which means that sometimes it is not easy to tell which action or sequence of actions are responsible for a particular positive reward that happens several time-steps later, this problem is known as credit assignment [34]. Besides rewards, if the environment provides some kind of rich and temporally dense multidimensional feedback, for example, measurements like kills, health, ammunition levels in a first-person shooter game, the agent can be programmed to learn to predict such rich and temporally dense measurements feedback instead [35]. It is possible for agents at inference time to observe the effects of different actions on such measurement streams and choose the action that maximizes an objective that can be expressed as a function of the predicted measurements at time 't' (i.e. m_t), for example, if the scenario(s) of the first-person shooter game (FPS) Doom is considered, and if the predicted measurements vector is (Kills, Ammo-used, Health) and the objective is to maximize the number of kills, then the objective can be set as,

$$U = f(m_t) = g \cdot m_t = 1 \times \text{Kills} - 0.5 \times \text{Ammo\_used} + 0.5 \times \text{Health} \qquad (1)$$

Where g = [1, -0.5, 0.5] is known as the goal vector. The - 0.5 weight assigned to the Ammo-used measurement just informs the agent it's not good to waste ammo. Such an approach has two major benefits that are as follows,

3.1. To Stabilize and Accelerate the Training

Dealing supervised learning [36], with concrete labels such as multidimensional measurements attached to each input state for example pixels' input, the agent is able to learn from a richer and denser signal than a single scalar reward stream can provide. Training performance can be greatly enhanced and stabilized as a result, just like in typical supervised learning tasks such as image classification.

3.2. Complex Goals at inference time

It is one of the more interesting aspects of this approach. In traditional reinforcement learning the objective is to maximize the expected future rewards, to be more specific, it implies that the agent only knows how to act based on the objective given. The agent cannot be simply instructed to behave differently (i.e. with another objective) in any meaningful sense at inference time [37].

In contrast, the presented supervised learning approach enables the agent to flexibly pursue different objectives (i.e. goals) or a combination of multiple objectives at inference time. It can be achieved through the model under supervised learning settings that outputs the prediction of measurements. The objective can be basically expressed as a function of the predicted measurements.

In an FPS game, the environment provides at least three measurements for every time step (Kills, Health, Health Packs). A health pack is a box scattered around the environment that can be picked up by the agent to improve its health. Health Packs measures the number of health packs picked up by the agent, to be more specific, it is a reasonable objective to simply tell the agent to maximize the number of kills [38],

$$U = 1 \times \text{kills} + 0 \times \text{Heath} + 0 \times \text{Health Packs} \qquad (2)$$

The coefficients of the measurements i.e. [1, 0, 0] represent the goal vector (g). Then at each time step, the agent will pick the action i.e. Turn Left, Turn Right, or Shoot that maximizes U, which is equivalent to saying pick the action which will lead to an increase in expected kill counts.

Further, if the health-level drops below a particular threshold, a different objective to the agent can be assigned so that it could focus on picking up health packs in order to improve health and avoid dying.

$$U = 0 \times \text{kills} + 0 \times \text{Heath} + 1 \times \text{Health Packs} \qquad (3)$$

Under this goal vector [0, 0, 1], the agent concentrates obsessively to pick up health packs. Once the health level goes back to normal, the goal vector can be switched back to [1, 0, 0] so that the agent could start killing again.

The ability to pursue complex goals at inference time has great implications for reinforcement learning, so a truly intelligent agent needs to be able to adapt itself to different goals under different circumstances [39]. However, these days most traditional reinforcement learning methods limit learning to only a single objective following the guidance of the scalar reward which is not the true way of learning for intelligent agents to behave. In fact, similar to human beings, reinforcement learning agents need to possess the innate ability to switch goals based on different circumstances [40].

As in traditional reinforcement learning, responses are received in the form of scalar rewards. In the same way, in DFP responses are received in the form of measurements (m) which can be thought of as a multidimensional vector with each element capturing some aspects of the game e.g. kills, ammunition, health, etc.,

Let $[\tau_1..., \tau_n]$ be a set of temporal offsets that the model has to learn to predict the differences between future and present measurements can be formulated as follows:

$$f = [m_{t+\tau 1} - m_t, ..., m_{t+\tau n} - m_t] \qquad (4)$$

It is beneficial to use [1, 2, 4, 8, 16, 32] as the set of temporal offsets. In practice, the model outputs a set of 'f', one for each action. At inference time, the agent simply picks the action that maximizes the objective U that can be computed as follows:

$$U = g \cdot f \qquad (5)$$

Here 'g' denotes the goal vector that controls the behaviour of the agent, the scalar reward is used as the only measurement and set 'g' as a vector of discounted factors i.e. $(1, \gamma, \gamma 2, ...)$ then the resulting objective function resembles the Q value, which is the sum of discounted future rewards. Therefore, sometimes in a sense, DQN is vaguely viewed and considered as a special case of DFP.

The network model used in the implementation consists of three inputs modules i.e. a perception module S(s), a measurement module M(m) and a goal module G(g) as shown in fig. 4. If 's' is an image, then the perception module 'S' is implemented as a convolutional neural network. The measurement and goal modules are fully-connected networks. The outputs of the three input modules are concatenated, forming the joint input representation (j) which is used for subsequent processing:

$$J = J\,(s, m, g) = [\ h\ S(s), M(m), G(g)] \qquad (6)$$

The model consists of two streams, the expectation stream E(j) and the Action Stream A(j). Usually using such two separate streams leads to better performance and this approach is based on the dueling architecture introduced by Google Deep Mind [41].

While training, each transition produces a tuple (s, a, m) using the environment. where ‘s’ represents the state e.g. image pixels, ‘a’ is the action taken, and ‘m’ is the measurement. A training target ‘f’ can then be formulated using the measurements obtained at the specified temporal offsets τ i.e. [1, 2, 4, 8, 16, 32],

$$f = [m_{t+\tau 1} - m_t, ..., m_{t+\tau n} - m_t] \quad (7)$$

The target can be used to train the neural network model with backpropagation and Mean Square Error (MSE) can be used as the loss function to compute the error.

3.3. Supervised Learning for Reinforcement Learning

The measurement and goal input modules are found less useful or in other words, found slightly detrimental to the performance so it is decided to use only the perception module as inputs to the model. The measurements are normalized and a goal vector ‘g’ of [1, 1, -1] i.e. coefficients for (Health, Health Packs, Poison) measurements are used. There are almost 50,000 episodes of DFP ran on the health gathering scenario.

DFP excels in environments where a stream of rich and temporally dense multidimensional feedbacks are available. In traditional reinforcement learning settings, transforming the feedbacks into a single dimension scalar reward might result in loss of useful information which would detriment performance. It is also noted out that enrichment in measurements is the most important factor for the good performance of DFP.

3.4. Using Health as the Measurement

Health is considered in the implementation, as health packs and Poison are derivative measurements from health. This is quite surprising and unreasonable that the performance of DFP deteriorates by 50% if just ‘health’ is used as the only measurement, even though health packs and poisons are derived from the change in health. Further, there is a beneficial effect by allowing the model to generate a richer set of predictions, similar to the way auxiliary tasks enhance the performance of deep learning vision classifier [42]. One of the best things about DFP is its capability to pursue different goals at inference time. For illustrating, the trained model can be used and the goal vector ‘g’ can be altered from (1, 1, -1) to (0, 0, 1). Where the objective becomes as below,

$$U = 0 \times \text{Health} + 0 \times \text{Health Packs} + 1 \times \text{Poison} \quad (8)$$

3.5. Experimental Setup

The experimental setup includes the environment and the architecture of the neural network explained as follows.

a) Neural Network Architecture

The neural network architecture consists of three input modules; the perception module is the environment state which is just screen pixels (input image). The three-layered convolutional neural network is used as the feature extractor to transform the screen pixels into a vector of length 512 and the three-layered fully connected network is used to parse the measurement module and goal module. The outputs of the three modules are concatenated to form a joint representation for further processing. The model is then split into two streams, the expectation stream, and the action stream. Their respective outputs are summed to form the model's prediction. In our proposed method the measurement size is three i.e. (health, health packs, poison), a number of time steps are 6 i.e. (1, 2, 4, 8, 16, 32) and the action size is three i.e. (Turn Left, Turn Right, and Move Forward). The model is trained and compiled using Adam optimizer [43] and the mean squared error (MSE) [44] as the loss metric. Similar to other reinforcement learning algorithms, most of the logic is contained in the update step. First, a mini-batch is sampled of sample trajectories from the experience replay buffer (memory) and initialize the corresponding states, measurements, goal and targets variables. Then target is computed for the model which is the difference between future and present measurements for the set of temporal offsets (1, 2, 4, 8, 16, 32). The target for each action is assigned to the 'f_action_target' variable. The rest of the variables are filled up with the mini-batch samples drawn from the experience replay. f_target is the ground truth label assigned to the model for training. Further, the training routine needs a call to perform gradient descent update. The overall learning and testing process using DFP lasted for 12 hours on a powerful GPU machine whose specifications are described in Table 1.

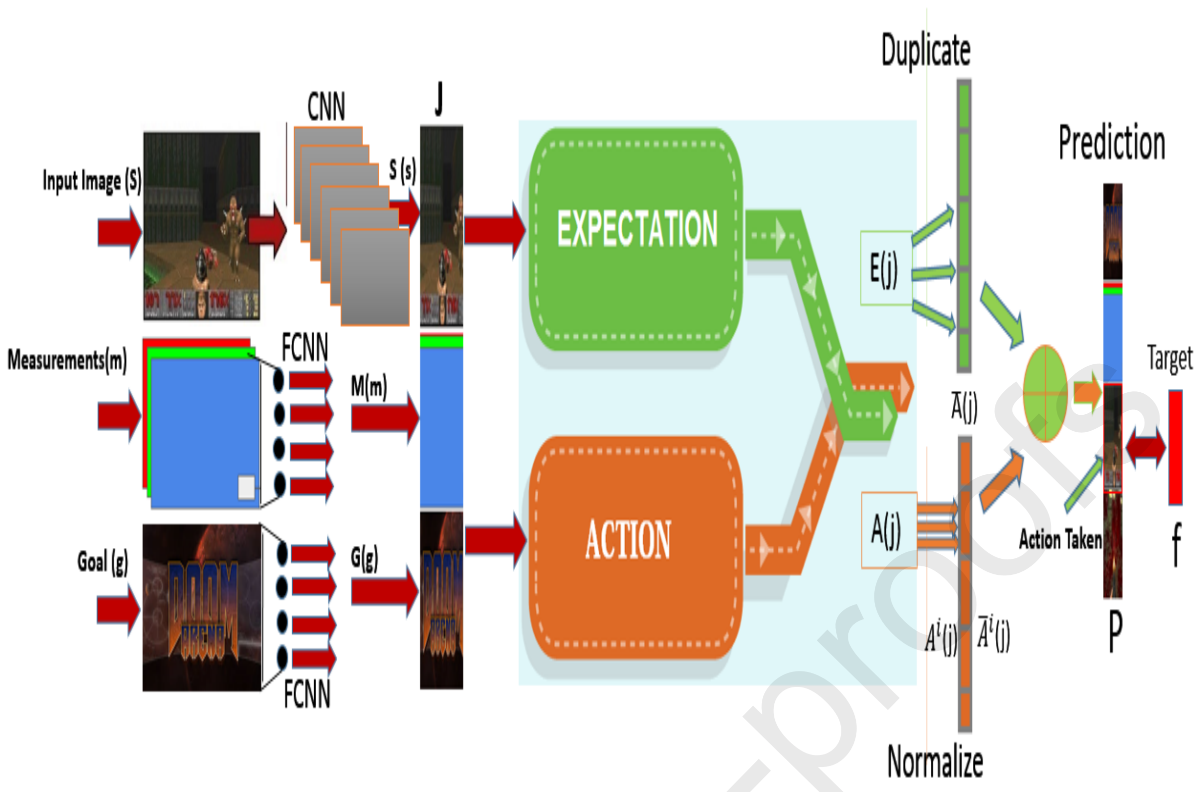

Fig. 4. DFP Neural Network Architecture

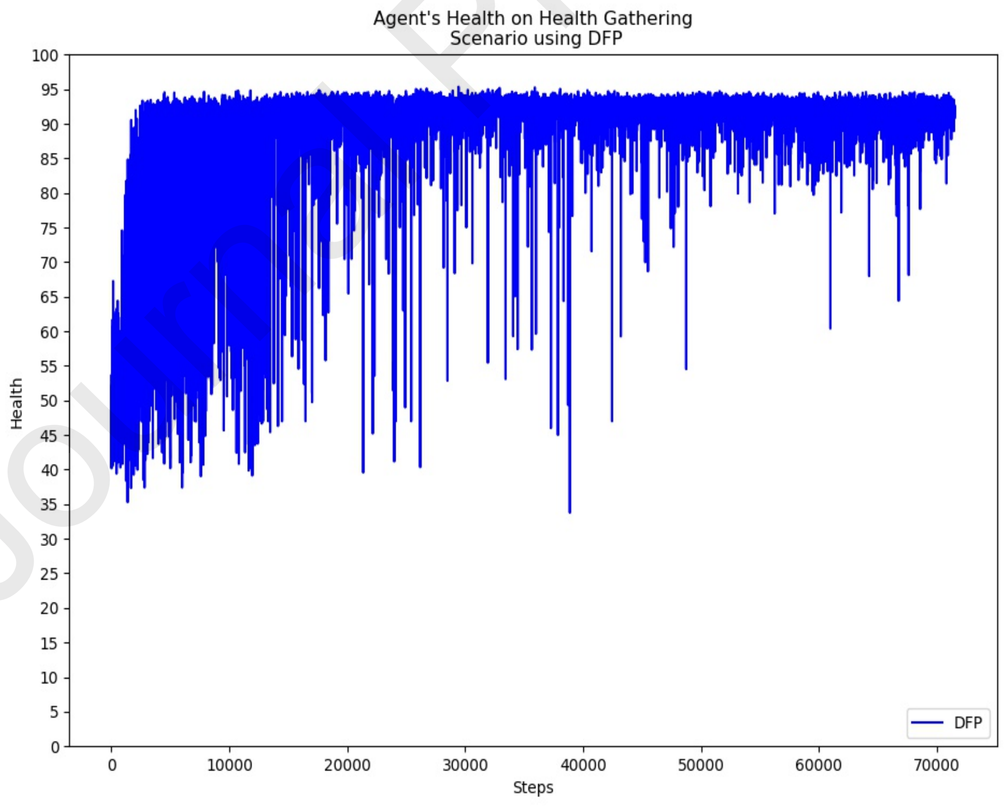

Fig. 5. The health of the agent trained with DFP

4. Direct Future Prediction (DFP) and Asynchronous Advantage Actor-Critic (A3C)

As the Direct Future Prediction (DFP) is explained in section 2 which is good at pursuing complex goals at test time. The architecture of DFP is shown in fig. 4 and its performance is presented in a graph displayed in fig. 5. With a performance of almost 95%, DFP can be chosen as one of the best machine learning technique for training game agents or bots. While on the other hand, the architecture of A3C utilizes the power of the deep neural networks (DNN) [45] by running multiple agents for training at the same time. Each agent then shares its results with the other agents. Since every agent makes different decisions, this approach reduces the chance for the AI to run into a local minimum. Additionally, it drastically reduces the average training time required to perform decently well at any given task. The A3C high-level architecture is shown in fig. 6. In A3C the global network and multiple worker agents each have their own set of network parameters. Each of these agents interacts with its own copy of the environment at the same time as the other agents are interacting with their environments. The reason this works better than having a single agent is that the experience of each agent is independent of the experience of the others. In this way, the overall experience available for training becomes more diverse. In addition to using the A3C algorithms for training the agent RMSProp [46] is used as an optimizer during the experiments.

After both DFP and A3C are used to train the agents on health gathering scenario of the VizDoom Game AI research platform, then the agents were tested as well to analyze and observe their performances where DFP was found better than A3C as shown in fig. 7. The agent trained with DFP gathered almost 95% health in the allotted time while the agent trained with A3C gathered almost 90% health.

The overall learning and testing process is measured in time and lasted for 12 hours on a powerful GPU machine.

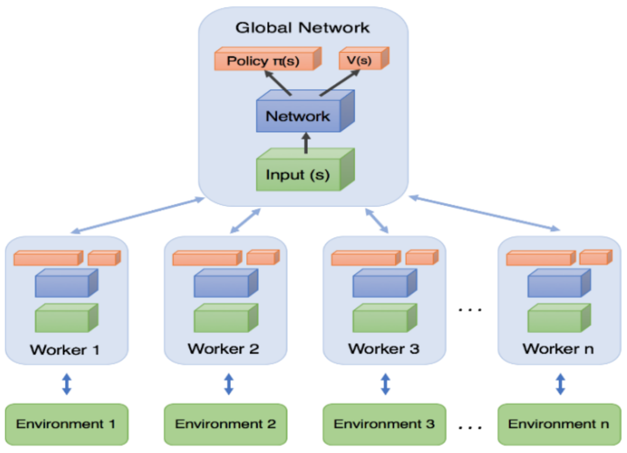

Fig. 6. Diagram of A3C high-level architecture [47]

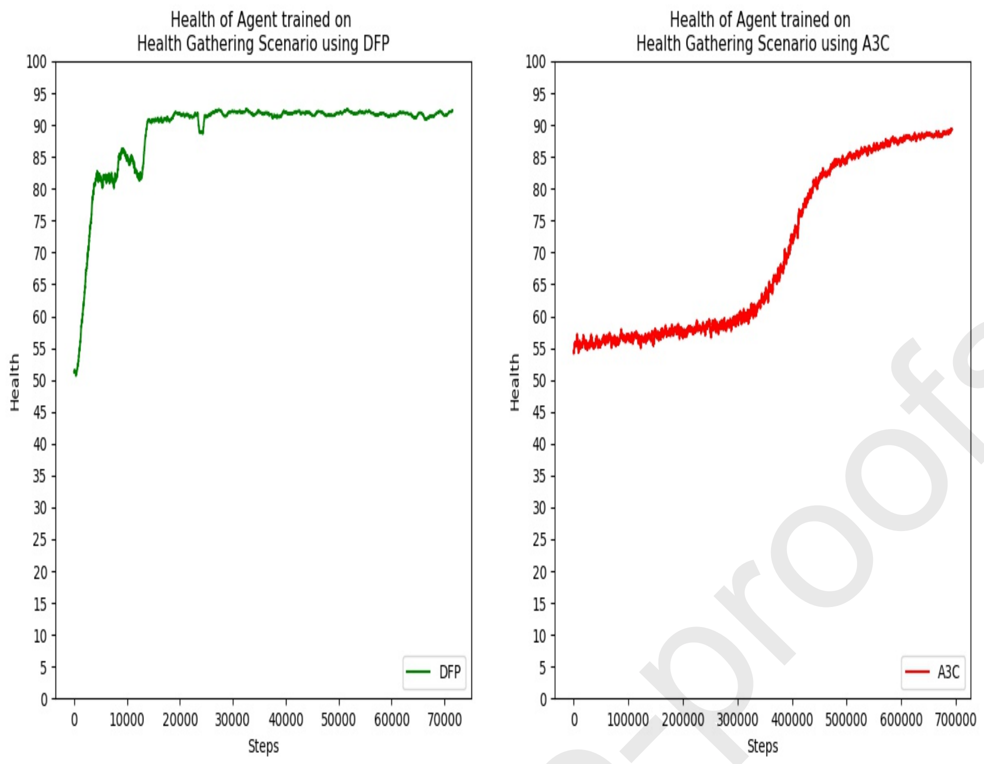

Fig. 7.Comparison of the performance of DFP with A3C on health gathering Scenario(s) using the VizDoom AI platform

5. Advantage Actor-critic (A2C) and Advantage Actor-Critic- Long Short-Term Memory (A2C-LSTM)

Advantage Actor-Critic (A2C) method performs best with large batch sizes by using the GPU's effectively. The A2C implementation is more cost-effective than A3C when using single-GPU machines, and is faster than a CPU-only A3C implementation when using larger policies, however, frailer in results and output than GPU-only A3C implementation. Both value-based methods and policy-based methods have drawbacks that's why a new reinforcement learning method 'Actor-Critic' has been introduced where critic measures how good the action taken is (value-based) and Actor controls how the agent behaves (policy-based). Mastering such an architecture is essential to understand the state of the art algorithms such as Proximal Policy Optimization (PPO) [48] which is based on Advantage Actor-Critic (A2C). In the Actor-Critic method, the actor makes actions randomly and the Critic observes the actions and provides feedback. Learning from this feedback, the actor updates its policy and becomes better at playing the game. On the other hand, the Critic also updates its knowledge to provide feedback so it can become better next time. The approach of Actor-Critic is to have two neural networks run in parallel which can be estimated as Actor: a policy function Π(s, a, Θ) that controls how the agent acts, and, the Critic: a value function $\hat{q}(s, a, w)$ measures how good the actions are.

In addition, as the value-based methods have high susceptibility so the advantage function can be used instead of value function to overcome this problem which can be defined as,

$$A(s, a) = Q(s, a) - V(s) \tag{9}$$

But even this advantage function has drawbacks because it requires two value functions-Q(s, a) and V(s) and this drawback can be overcome by using the TD error as a good estimator**.**

In addition, the agents trained with A2C and A2C-LSTM algorithms can be analyzed and observed in fig. 8, where the performance of A2C is better than A2C-LSTM. The curve of the A2C-LSTM stayed uniform until the last higher number of steps and never rose or declined to a high extent. While on the other hand, the beginning health gathering performance of the agent trained with A2C was found at most 2 to 3 percent better than A2C-LSTM and remain uniform until 10,000 steps where then the agent health gathering performance improved gradually stepwise and at ~15000 steps the performance curve started touching 70% but its overall final performance declines to ~54% which can be seen in the left graph in fig. 8. However, on the other side, to confirm the final conclusion to be accurate and authentic the A2C-LSTM agent was considered to be trained time and time again for further longer steps at least up to 300,000 steps which in number were more steps than the training steps of A2C agent just because to see any change or improvement in performance curve, however, despite of training for extraordinary steps still the curve was uniform. Therefore, it was concluded that A2C should be preferred over A2C-LSTM for training game agents or bots.

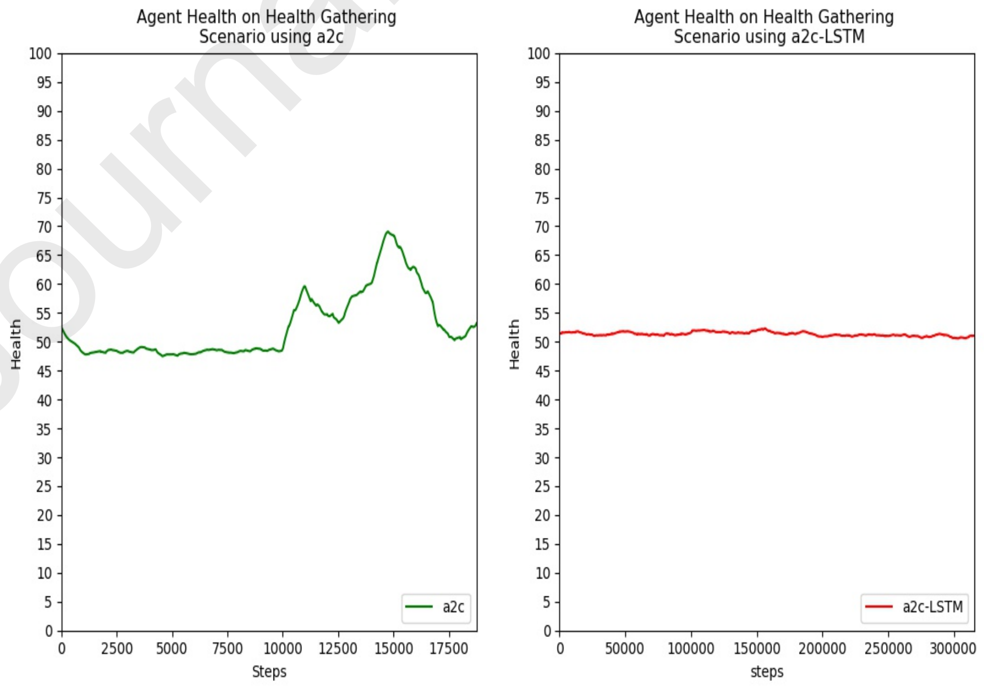

Fig. 8. Comparison of the performance of A2C with A2C-LSTM on health gathering Scenario(s) using the VizDoom AI platform

7. Asynchronous Advantage Actor-Critic (A3C) and Deep Recurrent Q-Network (DRQN)

The advantage of A3C over DRQN is that it is more resource-efficient, since A3C can be run on multiple cores of a single machine, and does not require a large amount of RAM to store the replay buffer compared to DRQN which requires a large amount of replay buffer. The actor-critic aspect provides more accurate updates to the policy than a DQN update might provide. A3C is an on-policy algorithm that cannot explore the state-space as efficiently as DQN, so there are some trade-offs between the two algorithms. However, according to our experiments conducted on the health gathering scenario of the VizDoom platform A3C performs higher than DRQN as can be seen in fig. 9. The DQN performs well than DRQN on fully observable environments (FOE) such as health gathering scenario shown in fig. 11, however, it performs low on partially observable environments (POE) where for overcoming this issue the recurrent feature (LSTM) of DRQN was introduced which makes it efficient over DQN by exploiting the experiences or sequential updates from memory however this feature does not make DRQN efficient over A3C algorithm and yet its performance is lower than A3C in many game environments.

The left graph in fig. 9 shows the performance of A3C which was not impressive initially until around 375,000 steps. however, Later the performance improved but it took significantly longer to achieve ~90% results while on the other hand, the right graph shows the steady performance of DRQN on the health gathering scenario which remains almost steady and uniform. DRQN never achieved results to a great extent and its performance crest and trough always remain between ~50 % and ~51%.

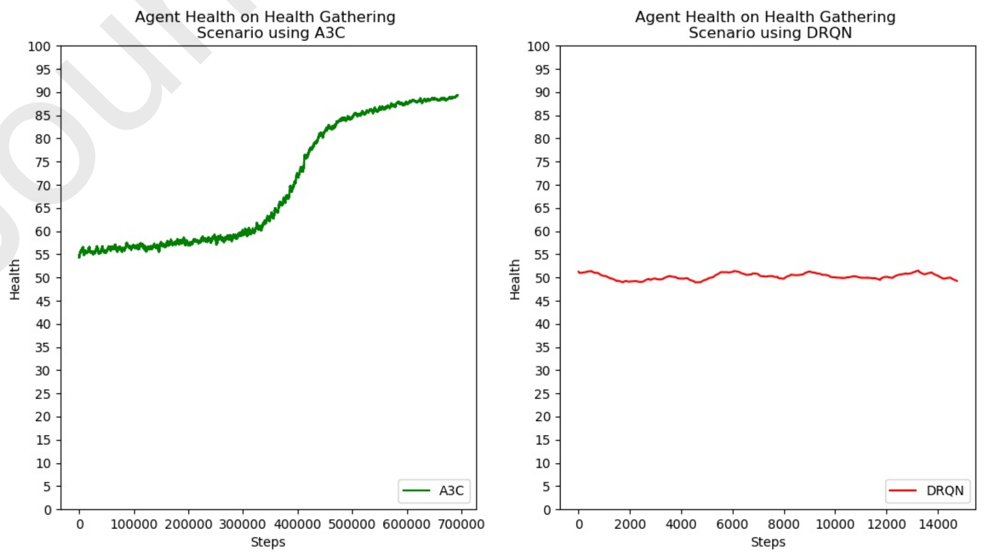

Fig. 9. Comparison of the performance of A3C with DRQN on health gathering Scenario(s) using the VizDoom AI platform

8. Direct Future Prediction (DFP) and Deep Recurrent Q-Network (DRQN)

In order to compare the performance of DFP with DRQN, two agents are trained on the VizDoom health gathering scenario using DFP and DRQN, then both the agents are tested on the health gathering scenario (maps) to see their performance difference where DFP outperformed DRQN as shown in fig. 10. While gathering health packs the agent trained with DRQN was losing health on average and never manage to improve it to a high level. Thus its overall final performance remains average with ~50% to 51% as shown in the right-side graph. On the other hand, the agent trained with DFP performed very well in collecting health packs and the overall health of the agent achieved ~94% results which can be observed in the left graph for further consideration. It concludes that DFP is a better technique which is one among the state-of-the-art for training agents using the VizDoom Game-AI research platform.

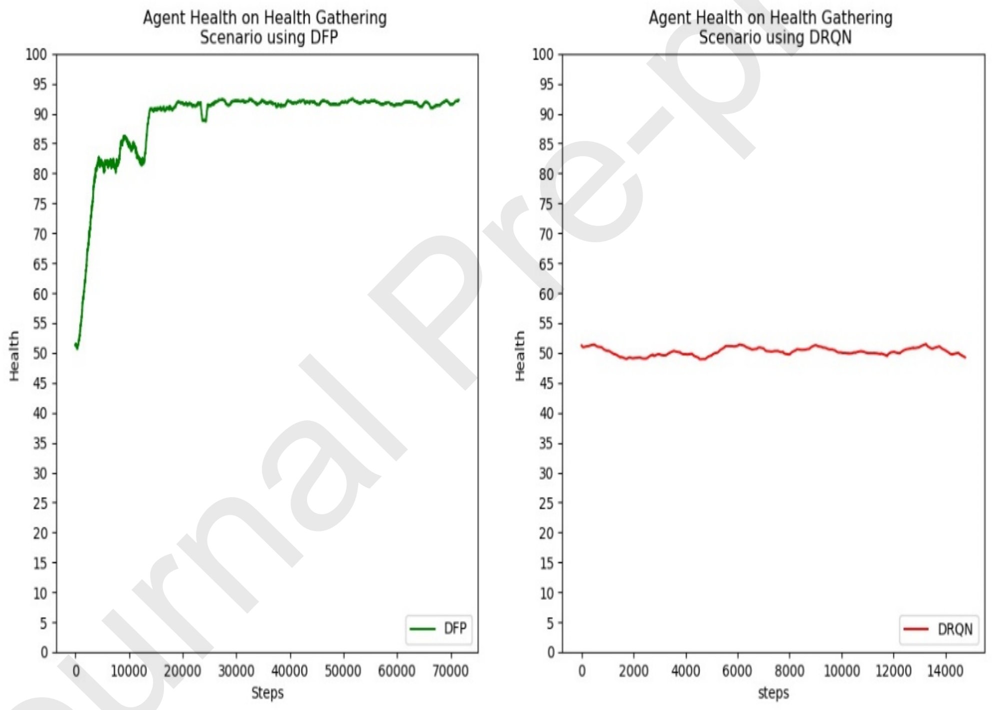

Fig. 10. Performance Comparison of DFP with DRQN on health gathering Scenario(s) using the VizDoom AI platform

9. Deep Q-Network (DQN) and Deep Recurrent Q-Network (DRQN)

In DQN, a single agent is represented by a single neural network that interacts with a single environment. Deep Q-Networks are more capable of overcoming unstable learning by mainly 4 techniques i.e. Experience Replay [49], Target Network, Clipping Rewards and Skipping Frames.

Experience reply was originally proposed in 1993 in [50] to overcome the problem of overfitting as Deep Neural Networks (DNN) easily overfits current episodes and once the DNN gets over fit then it becomes hard to produce different experiences. So, for solving this issue, experience replay stores experiences including state transitions, rewards, and actions, which are necessary data to perform Q learning, and makes mini-batches to update neural networks. While calculating the temporal difference (TD) error the target Q-function (Target network) gets changed frequently with DNN's as unstable target Q-function makes training difficult so target network technique fixes parameters of target Q-function Q (s_{t+1}, ʽP) and replaces them with the latest network every thousand of steps.

$$Q(s_t\,, a) \leftarrow Q(s_t\,, a) + \alpha\,[r_{t+1} + \gamma \max_{p} Q(s_{t+1}\,, {}^{\prime}P) - Q(\,s_t, a)] \qquad (10)$$

Each computer game has different score scales. For example, in Pong, players can get 1 point when winning the play. Otherwise, players get -1 point. However, in Space Invaders, players get 10~30 points when defeating invaders. This difference would make training unstable. Thus Clipping Rewards technique clips scores, which all positive rewards are set +1 and all negative rewards are set -1.

The fourth technique of skipping frames that DQN uses to overcome unstable learning can be explained excellently by referring or with the help of the arcade learning environment (ALE) [51] which is capable of rendering 60 images per second. But actually, players don't take actions thus much in a second. AI doesn't need to calculate Q-values every frame. So by employing skipping frames technique DQN calculates Q-values every 4 frames and uses past 4 frames as inputs. This reduces the computational cost and gathers more experiences.

Using these four techniques enables DQN to achieve stable training. Table 3 shows that the performance of DQN increases if it uses Experience Replay technique along with the Target Network.

Table 3. DQN Performance with and without Experience Replay and Target Network [52]

Replay	✓	✓	✗	✗
Target	✓	✗	✓	✗
Breakout	**316.8**	240.7	10.2	3.2
River Raid	**7446.6**	4102.8	2867.7	1453.0
Sequest	**2894.4**	822.6	1003.0	275.8
Space Invaders	**1088.9**	826.3	373.2	302.0

DQN and DRQN are the variants of Deep Q-learning introduced by Google DeepMind, London, the UK in 2013 and 2015 that performed with extraordinary results on Atari games and later were applied to different platforms. In this paper, we also applied them using the VizDoom AI platform to compare and differentiate their performance on the health gathering scenario (map) by training the agents. The agent trained with DQN performed slightly better than the DRQN in gathering the health packs due to the fully observable environment and retained its health to ~53%. while on the other hand, the agent trained with DRQN remain below in performance in collecting health packs on the health gathering scenario (maps) by retaining its health to almost 51% as shown in fig. 11. It concludes that the agents trained with DQN perform better than DRQN particularly on the health gathering scenario of the VizDoom AI platform.

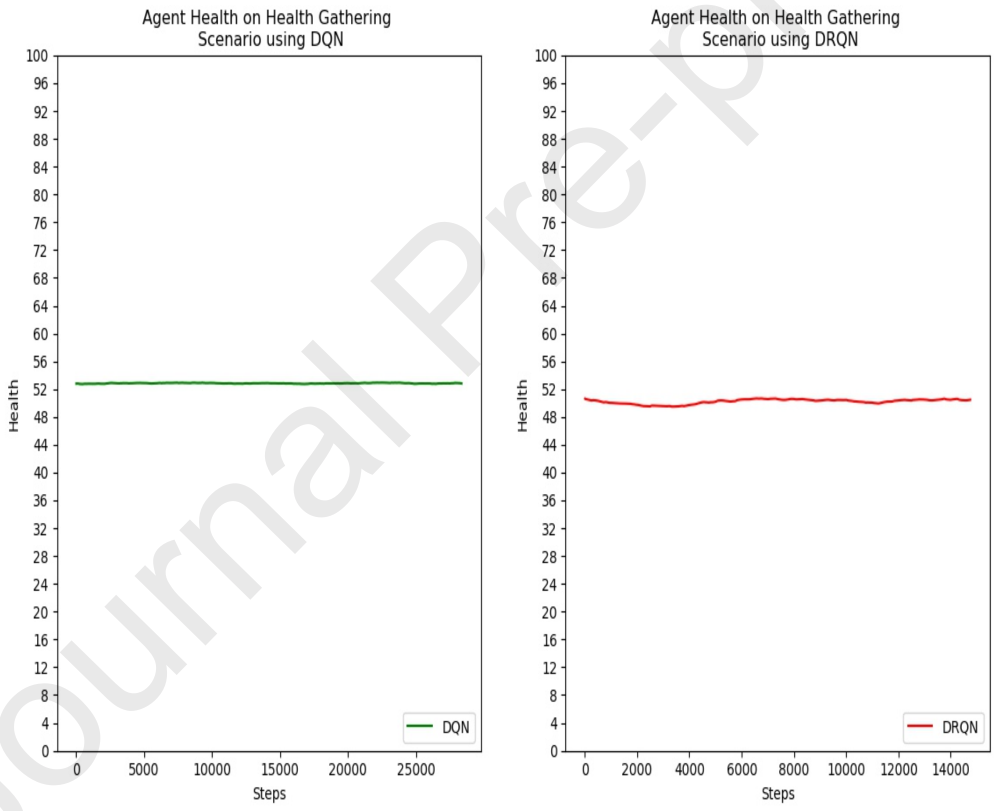

Fig. 11. Comparison of the performance of DQN with DRQN on health gathering Scenario(s) using the VizDoom AI platform

10. Dueling Deep Q-Network (DDQN) and Double Deep Q-Network (DDQN)

Dueling-DQN (DDQN) and double-DQN (DDQN) are two simple additional improvements to the DQN architecture or, in other words, these are close newer variants of DQN introduced by Google DeepMind, London, UK, in 2015-16 [17, 53]. The dueling-DQN and double-DQN allow for improved performance, stability, and faster training time. The double-DQN basically uses 2 neural networks to perform the Bellman iteration, one for generating the prediction term and the other for generating the target term. It helps to alleviate the bias introduced by the inaccuracies of Q-network at the beginning phase of training.

The regular DQN often overestimates the Q-values of the potential actions to take in a given state. While this would be fine if all actions were always overestimated equally, but this never finds to be the case because if certain suboptimal actions regularly were given higher Q-values than optimal actions, the agent would have a hard time ever learning the ideal policy. In order to fix this issue, a simple trick was proposed: instead of taking the max over Q-values when computing the Target Q-value for training steps, the primary network is used to choose an action, and the target network is used to generate the target Q-value for that action. By decoupling the action choice from the target Q-value generation, it was able to substantially reduce the overestimation, and train faster and more reliably. The new double-DQN equation for updating the target value could be represented as,

$$\text{Q-Target} = r + \gamma\, Q\,(s, \text{argmax}(\,Q(s, a, \Theta)), \acute{\boxed{?}}) \qquad (11)$$

And, the advantage is calculated separately and then combined only at the final layer into a Q-value [17].

The reason behind the change in the architecture that dueling-DQN makes is to have a network that separately computes the advantage and value functions, and combines them back into a single Q-function only at the final layer.

$$Q\,(s, a) = V(s) + A(a) \qquad (12)$$

In dueling DQN, Q can also be computed by using the below formula with value function V and a state-dependent action advantage function A,

$$Q(s,a) = v(s) + A(s,a) - \frac{1}{|A|}\sum_{a=1}^{|A|} A(s,a)$$

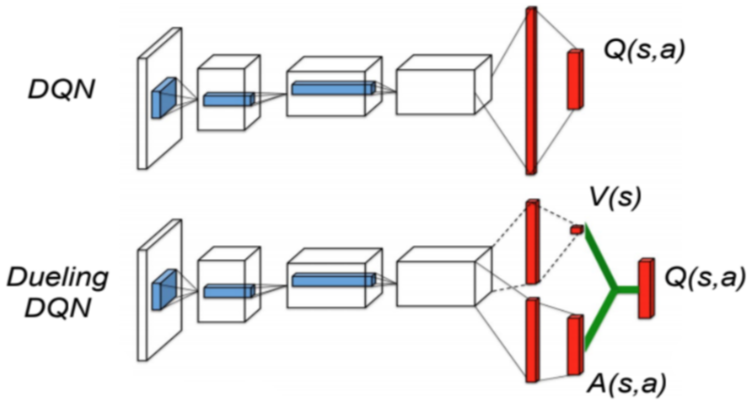

Fig. 12. Above: Regular DQN with a single stream for Q-values. Below: Dueling DQN where the value

So, dueling-DQN and double-DQN are the two variants of DQN that performed well on Atari 2600 domain. In this paper, we use them particularly using the VizDoom AI platform to compare and differentiate their performance on the health gathering scenario (map) by training two agents. The agent trained with dueling-DQN performed a lot better than the double-DQN in gathering the health packs and retained its health to ~59%. On the other hand, the agent trained with double-DQN remain below in performance by collecting health packs on the map and its health retains to ~52% as shown in fig.13. It concludes that the agents trained with dueling-DQN perform better particularly on the VizDoom AI platform.

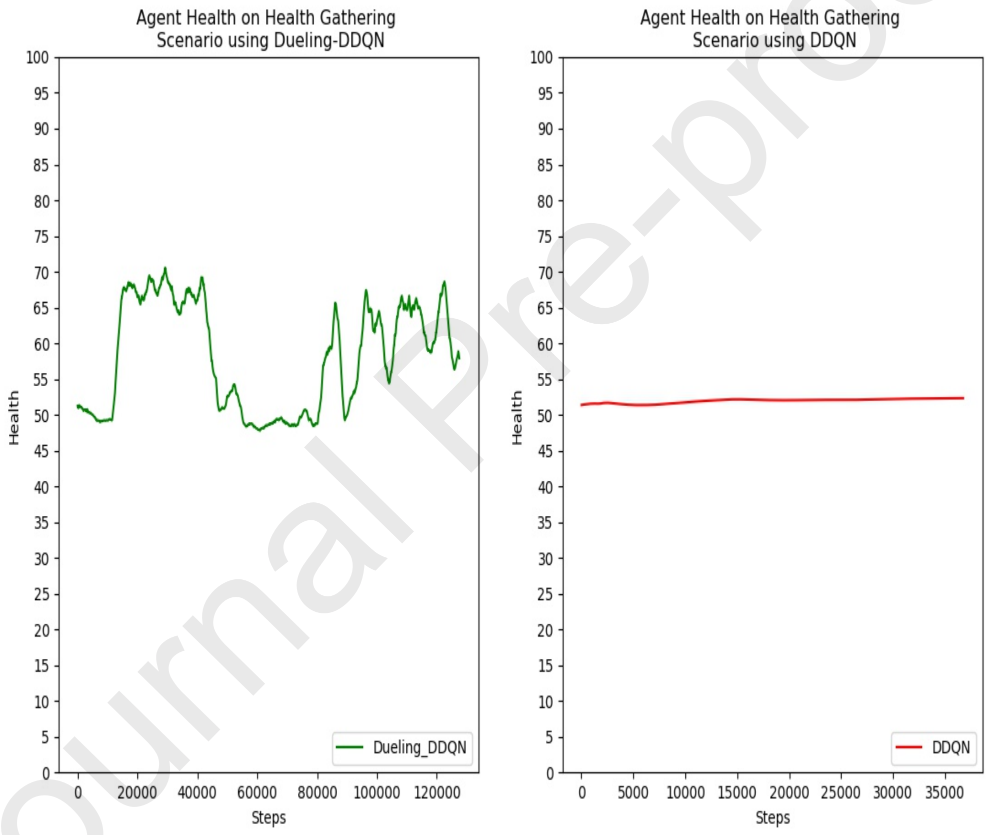

Fig. 13. Performance of Dueling-DQN with Double-DQN on health gathering Scenario(s) using the VizDoom AI platform

11. C51_DDQN and Reinforce

C51 is a viable algorithm introduced in [18] to perform an iterative approximation of the value distribution Z using distributional Bellman equation. The number 51 represents the use of 51 discrete values to parameterize the value distribution Z. During each update step, a transition is sampled from the environment and the target distribution $\acute{Z}$ is computed. $\acute{Z}$ is used to update the current distribution Z by minimizing the cross entropy loss between Z and $\acute{Z}$. The pseudo code of the C51 Algorithm is as follows.

Pseudo code of the C51 Algorithm [54]

```
**Input** a transition $x_t, a_t, r_t, x_{t+1}, \gamma_t \epsilon[0,1]$
$Q(x_{t+1}, a) := \sum_i z_i p_i(x_{t+1}, a)$
$a^* \leftarrow argmax_a Q(x_{t+1}, a)$
$m_i = 0, i \in 0, \ldots, N-1$
**For** $j \in 0, \ldots, N-1$ **do**
    # Compute the projection **of** $\hat{T}z_j$ onto the support $\{\hat{T}z_i\}$
    $\hat{T}z_j \leftarrow [r_t + \gamma_t z_j]_{V_{MIN}}^{V_{MAX}}$
    $b_j \leftarrow (\hat{T}z_j - V_{MIN})/\Delta z$
    $\iota \leftarrow \lfloor b_j \rfloor, u \leftarrow \lceil b_j \rceil$
    # Distribute probability of $\hat{T}z_j$
    $m_\iota \leftarrow m_\iota + p_j(x_{t+1}, a^*)(u - b_j)$
    $m_u \leftarrow m_u + p_j(x_{t+1}, a^*)(b_j - \iota)$
**End for**
**Output ---** $\sum_i m_i \log p_i(x_t, a_t)$ # $Cross - entropy\ loss$
```

Similar to DQN, we first use a deep neural network to represent the value distribution Z. Since the inputs are screen pixels, the first 3 layers are convolutional layers. The neural network outputs 3 sets of value distribution predictions, one for each action i.e. (Health, Health Packs, Poison). Each set of prediction is a softmax layer with 51 units and the number of atoms is the number of discrete values (i.e. 51). The main logic of the algorithm is contained in the update step. First, a minibatch of sample trajectories is sampled from the Experience Replay buffer and the corresponding states, reward, and targets variables are initialized. The variable ‘m_prob’ stores the probability mass of the value distribution Z. Next, a forward pass is carried out to get the next state distributions. As the model outputs 3 sets of value distributions, one for each action. Thus the one with the largest expected value to perform the update (similar to $\max_{a'} Q(s', a')$ in Q Learning) is only needed. Then the target distribution Z' is computed (i.e. scale by γ and shift by reward r) and projected it to the 51 discrete supports.

The C51 significantly outperforms several state-of-the-art algorithms. In fact, C51 surpasses much state-of-the-art by a large margin in a number of games, most notably Seaquest [51].

On the other hand, 'Reinforce' is a family of reinforcement learning methods which directly update the policy weights or in other words, it is a policy gradient-based method which samples the expected return directly from the episode where the expected return is actually the total episodic reward onward that step G_t. To be more specific, it iteratively updates agent's parameters by computing policy gradient and works well when episodes are reasonably short where lots of episodes can be simulated however value-function methods are better for longer episodes because they can start learning before the end of a single episode. How this algorithm works can be understood further with the help of the pseudocode as follows?

Pseudocode of the Reinforce Algorithm [18]

```
initialize $\theta$
for each episode $\{s_1, a_1, r_2 ... s_{T-1}, a_{T-1}, r_T\}$ s sampled from policy $\pi_\theta$ do
    for t = 1 to T - 1 do
        $\theta \leftarrow \theta + \alpha \nabla_\theta \log\pi_\theta (s_t, a_t) G_t$
    end for
end for
```

However, 'Reinforce' suffers from high variance because the sampled rewards can be different from one episode to another that is why this algorithm is normally used with a baseline subtracted from the policy.

In the same way, C51_DDQN and Reinforce are as well implemented and tested on the VizDoom health gathering scenario (maps). The agent trained with C51_DDQN performed a lot better and uniform than the Reinforce in gathering the health packs and retained its health to ~87%. On the other hand, the agent trained with Reinforce remains below in performance by collecting health packs on the health gathering scenario (maps) and retains its health finally to ~57% as its performance crest and trough can be observed in fig. 14. It concludes that the agents trained with C51_DDQN perform better than the agents trained with Reinforce particularly using the VizDoom Game-AI research platform.

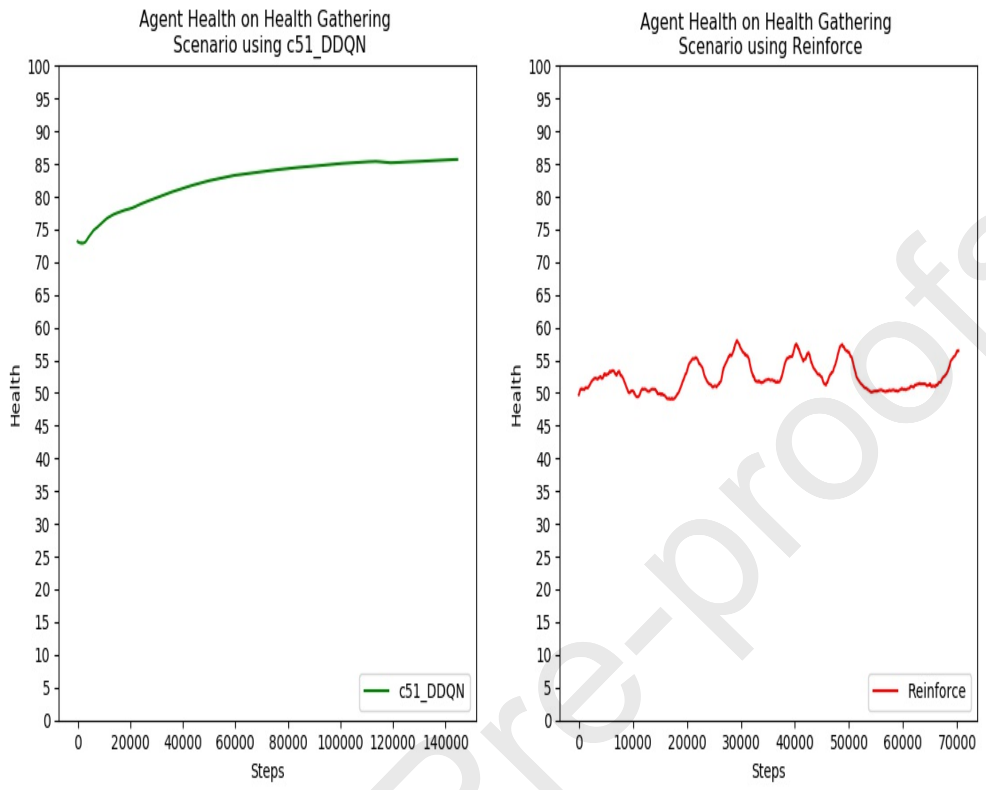

Fig. 14. Performance of C51_DDQN with Reinforce on health gathering Scenario(s) using the VizDoom AI platform

12. Techniques with Better Performance

After training agents on health gathering scenario(s) of the VizDoom AI platform using different machine learning methods and techniques, the performance of the following four methods is found better and accurate as shown in fig. 15. The x-axis represents the training steps of thousands to million while the y-axis represents the health of the agents in percentage while gathering the health packs. The agents trained using the DFP method sustains almost 95% health while gathering health packs which is more than other methods and remains one of the best options for training the agents. The agents trained using A3C method follow DFP in performance and sustains almost 90% health on health gathering scenarios which means the second-best machine learning method after DFP for training agents, However, it takes A3C significantly longer to achieve those results and is performing worse than the others until around 375,000 steps. The C51_DDQN is the third-best machine learning technique that performed well in gathering health on the health gathering scenario of the VizDoom Game-AI platform with ~87% performance. Dueling DQN (DDQN) is the last method among the four best machine learning methods that perform well in gathering health packs on the health gathering scenarios and retained health to ~59%.

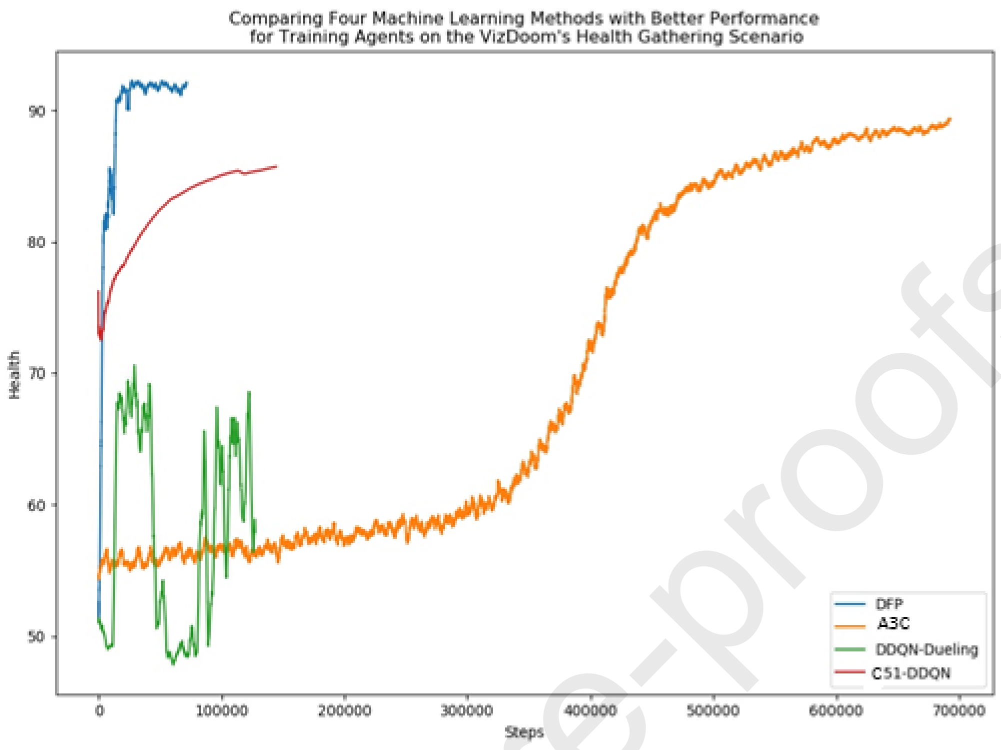

Fig. 15. Comparison of the four techniques with the best performance on the health gathering Scenario(s) of the VizDoom AI platform

13. Discussion and Future Work

In this paper, the work has shown the effectiveness of using different machine learning techniques and methods within the context of complex 3D multi-agent environment with agents playing competitively against human players and inbuilt game agents using the VizDoom Game AI research platform. Such systems can be applied to commercial games to provide competent opponents, without the need for predefined manual coded instructions or scripts. In addition, this paper concludes that if the environment provides with a rich and temporally dense measurements signals, reformulating the reinforcement learning problem to supervised learning (e.g. DFP) leads to better performance and accelerated training. According to the experiments, more measurements certainly lead to better results. There is a beneficiary effect by allowing the model to generate a richer set of predictions, similar to the way auxiliary tasks enhance the performance of deep learning vision classifier. The goal skeptical nature of DFP allows the agent to pursue complex goals at inference time. This lifts the limitation on learning and acting from a single objective by traditional reinforcement learning methods, and is one way to achieve transfer learning and one-shot learning for multiple tasks. Besides DFP, this paper as well concludes that machine learning techniques such as DFP, A3C, C51_DDQN, and

Dueling DQN can be chosen as the first choice for training agents due to their efficient performance. The insightful summary of what the experiment results convey is presented in table 4.

Table 4. Showing the results or performance of Algorithms used for training agents

Algorithms/Methods									
DFP	**A3C**	**A2C**	**A2C-LSTM**	**DQN**	**DRQN**	**Dueling DQN**	**Double DQN**	**C51_DDQN**	**Reinforce**
~95%	~90%	~54%	~52%	~53%	~51%	~59%	~52%	~87%	~57%

The experiments demonstrate that learning from raw pixels is a new era in artificial intelligence since of versatile and dynamic environments such as VizDoom. Extending such approaches further in multiple ways to broaden the range of behaviours for learning is our future work.

Acknowledgment

This work is funded by the University of Peshawar, Pakistan and the Higher Education Department KPK, Pakistan. The authors would also like to acknowledge and thank NVIDIA Corporation for two powerful sets of GPU machines donation.

Conflict of Interest

The authors declare that they have no conflicts of interest.

Data Availability

The research data (raw and processed) used to support the findings of this research are available from the corresponding author upon request.

References

[1]. Owens, J.D., et al., GPU computing. Proceedings of the IEEE, 2008. 96(5): p. 879-899.
[2]. Jouppi, N.P., et al. In-datacenter performance analysis of a tensor processing unit. in Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on. 2017. IEEE.
[3]. Hagan, M.T., et al., Neural network design. Vol. 20. 1996: Pws Pub. Boston.
[4]. Khan Adil, F.J., Shaohui Liu, Worku Jifara, Zhihong Tian, Yunsheng Fu, State-of-the-Art and Open Challenges in RTS Game-AI and Starcraft. (IJACSA) International Journal of Advanced Computer Science and Applications, 2017. 8(12): p. 9.
[5]. Mnih, V., et al., Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.
[6]. Lowney, M. and R. Mahieu, Creating an Agent of Doom: A Visual Reinforcement Learning Approach.
[7]. Pang, Z.-J., et al., On Reinforcement Learning for Full-length Game of StarCraft. arXiv preprint arXiv:1809.09095, 2018.
[8]. Grand theft auto V. 2014, Xbox One. New York, NY: Rockstar Games, [2014] ©2014.
[9]. Ekaputra, G., C. Lim, and K.I. Eng, Minecraft: A game as an education and scientific learning tool. ISICO 2013, 2013. 2013.
[10]. Grondman, I., et al., A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012. 42(6): p. 1291-1307.

[11]. Wang, J.X., et al., Learning to reinforcement learn. arXiv preprint arXiv:1611.05763, 2016.
[12]. Mnih, V., et al. Asynchronous methods for deep reinforcement learning. in International Conference on Machine Learning. 2016.
[13]. Freitas, S., et al., Exploration of DQN in ViZDoom. 2018.
[14]. Brejl, R., H. Purwins, and H. Schoenau-Fog, Exploring Deep Recurrent Q-Learning for Navigation in a 3D Environment. Eai Endorsed Transactions on Creative Technologies, 2018.
[15]. Schulze, C. and M. Schulze, ViZDoom: DRQN with Prioritized Experience Replay, Double-Q Learning, & Snapshot Ensembling. arXiv preprint arXiv:1801.01000, 2018.
[16]. Dabney, W., et al., Distributional reinforcement learning with quantile regression. arXiv preprint arXiv:1710.10044, 2017.
[17]. Wang, Z., et al., Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581, 2015.
[18]. Williams, R.J., Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 1992. 8(3-4): p. 229-256.
[19]. Kempka, M., et al., ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. arXiv preprint arXiv:1605.02097, 2016.
[20]. Khan, A., et al., Optimal Skipping Rates: Training Agents with Fine-Grained Control Using Deep Reinforcement Learning. Journal of Robotics, 2019. 2019: p. 10.
[21]. Stooke, A. and P. Abbeel, Accelerated methods for deep reinforcement learning. arXiv preprint arXiv:1803.02811, 2018.
[22]. Lample, G. and D.S. Chaplot, Playing FPS games with deep reinforcement learning. arXiv preprint arXiv:1609.05521, 2016.
[23]. Khan Adil, F.J., Shaohui Liu, Aleksei Grigorev, B. B. Gupta, Seungmin Rho, Training an Agent for FPS Doom Game using Visual Reinforcement Learning and VizDoom. (IJACSA) International Journal of Advanced Computer Science and Applications, 2017. 8(12).
[24]. Watkins, C.J. and P. Dayan, Q-learning. Machine learning, 1992. 8(3-4): p. 279-292.
[25]. Dosovitskiy, A. and V. Koltun, Learning to act by predicting the future. arXiv preprint arXiv:1611.01779, 2016.
[26]. LEUNG, M.H. and L.R.T. Michael, Applying Modern Reinforcement Learning to Play Video Games. 2017.
[27]. Hochreiter, S. and J. Schmidhuber, Long short-term memory. Neural computation, 1997. 9(8): p. 1735-1780.
[28]. Silver, D., et al., Mastering the game of Go with deep neural networks and tree search. Nature, 2016. 529(7587): p. 484-489.
[29]. Nachum, O., et al. Bridging the gap between value and policy based reinforcement learning. in Advances in Neural Information Processing Systems. 2017.
[30]. Kassambara, A., Machine Learning Essentials: Practical Guide in R. 2018: STHDA.
[31]. Abadi, M., et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467, 2016.
[32]. Bradski, G. and A. Kaehler, OpenCV. Dr. Dobb's journal of software tools, 2000. 3.
[33]. Fairbank, M. and E. Alonso, The Divergence of Reinforcement Learning Algorithms with Value-Iteration and Function Ap-proximation. arXiv preprint arXiv:1107.4606, 2011.
[34]. Glavin, F.G. and M.G. Madden, Learning to Shoot in First Person Shooter Games by Stabilizing Actions and Clustering Rewards for Reinforcement Learning. arXiv preprint arXiv:1806.05117, 2018.
[35]. Hafner, D., Deep Reinforcement Learning From Raw Pixels in Doom. arXiv preprint arXiv:1610.02164, 2016.
[36]. Smola, A. and S. Vishwanathan, Introduction to machine learning. Cambridge University, UK, 2008. 32: p. 34.
[37]. McPartland, M. and M. Gallagher. Creating a multi-purpose first-person shooter bot with reinforcement learning. in Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On. 2008. IEEE.
[38]. Wydmuch, M., M. Kempka, and W. Jaśkowski, ViZDoom Competitions: Playing Doom from Pixels. arXiv preprint arXiv:1809.03470, 2018.

[39]. Pérez-Liébana, D., et al. Analyzing the robustness of general video game playing agents. in Computational Intelligence and Games (CIG), 2016 IEEE Conference on. 2016. IEEE.
[40]. Glavin, F.G. and M.G. Madden. DRE-Bot: A hierarchical First Person Shooter bot using multiple Sarsa (λ) reinforcement learners. in 2012 17th International Conference on Computer Games (CGAMES). 2012. IEEE.
[41]. Powles, J. and H. Hodson, Google DeepMind and healthcare in an age of algorithms. Health and Technology, 2017. 7(4): p. 351-367.
[42]. LeCun, Y., Y. Bengio, and G. Hinton, Deep learning. Nature, 2015. 521(7553): p. 436-444.
[43]. Kingma, D.P. and J. Ba, Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
[44]. Das, K., J. Jiang, and J. Rao, Mean squared error of empirical predictor. The Annals of Statistics, 2004. 32(2): p. 818-840.
[45]. Schmidhuber, J., Deep learning in neural networks: An overview. Neural networks, 2015. 61: p. 85-117.
[46]. Ruder, S., An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747, 2016.
[47]. Tan, B., N. Xu, and B. Kong, Autonomous Driving in Reality with Reinforcement Learning and Image Translation. arXiv preprint arXiv:1801.05299, 2018.
[48]. Schulman, J., et al., Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
[49]. Schaul, T., et al., Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015.
[50]. Lin, L.-J., Reinforcement learning for robots using neural networks. 1993, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.
[51]. Bellemare, M.G., et al., The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 2013. 47: p. 253-279.
[52]. Mnih, V., et al., Human-level control through deep reinforcement learning. Nature, 2015. 518(7540): p. 529-533.
[53]. Silver, H.v.H.a.A.G.a.D., Deep Reinforcement Learning with Double Q-learning.pdf. Association for the Advancement of Artificial intelligence, 2016.
[54]. Bellemare, M.G., W. Dabney, and R. Munos, A distributional perspective on reinforcement learning. arXiv preprint arXiv:1707.06887, 2017.

Corresponding Author Profile

Adil Khan

http://orcid.org/0000-0003-2862-5718 (ORCID ID)

He received PhD in Artificial Intelligence from the University of Peshawar, M.S in Computer Science from City University of Science and Information Technology Peshawar, BS Honors in Computer Science from Edwards college Peshawar, B. Ed from the University of Peshawar and C.T. from AIOU Islamabad, Pakistan. From November 2014 to August 2016, he was a Lecturer in Higher Education Department KP, Pakistan and from September 2016 to August 2019 he was a research scholar in Game-AI at the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001 PR China. Currently, he is working as an Assistant Professor at the University of Peshawar, Peshawar, Pakistan. He has published many publications in top-tier academic journals and conferences. Adil Khan is interested in Machine Learning, Neural networks, Game Artificial Intelligence (Game-AI), Real-Time Strategy Games, First-Person-Shooter Games, Sandbox Open World Games, Computer Vision and Image Processing (Breast Cancer Detection). He can be reached at personal E-mail: adil.adil25@yahoo.com

CONFLICT OF INTEREST

Authors declare that they have no conflict of interest or whatsoever

HIGHLIGHTS

\`          \`

1) Have shown the effectiveness of using different machine learning techniques and methods.

2) Machine learning is employed in video games to control non-human computer-players, agents or bots.

3) VizDoom', a new 3D partially observable environment introduced in recent past is used, with interesting

enhanced properties that make agents stand out from inbuilt AI agents and human players.

4) Most of the machine learning techniques were employed before in Atari Games that took place in 2D

environments that were not fully observable to the agents. However, in this paper, machine learning

techniques and methods are employed in 3D environments that are fully observable to the agents such as

Doom, a first-person-shooter (FPS) game.

5) Direct Future Prediction (DFP) is used which is one of the best technique for training agents.

6) The performance of the agents is compared and differentiated.