General Game Playing with Reinforcement Learning

Florida Atlantic University * Shaun Pritchard * Email:spritchard2021@fau.edu

*Published December 13, 2021 * Department of Computer Science and Engineering FAU*

2021

General Game Playing with Reinforcement Learning

Florida Atlantic University * Shaun Pritchard * Email:spritchard2021@fau.edu

Abstract -In computational science and robotics, game playing and artificial intelligence (AI) are research areas that have long been studied. RL systems represent a major step towards autonomous systems that comprehend the visual world at an incredibly deep level and are poised to revolutionize the field of artificial intelligence (AI). Game Play is the testbench where environmental variables can be evaluated to generate adaptive, intelligent, or responsive behavior or simulate real-world scenarios. In this survey, we will explore general gameplay (GGP) with reinforcement learning and its various applications. This survey will cover central algorithms in deep RL, including the deep Q-network (DQN) and general mathematical and pragmatic approaches taken in the field. The objective of the current review is to examine the history, associations, and recent advances in the field of general game playing with reinforcement learning and its subfields. We conclude by describing several current areas of research within this field.

Index Terms – Advances in reinforcement Learning, Artificial Intelligence , General Game Play reinforcement learning, Q-learning,

1 Introduction

Game playing and artificial intelligence (AI) are long-standing research areas in the field of computational science and robotics. A subset of artificial intelligence, reinforcement learning plays a crucial role in game playing and defining intelligent data models for real-world applications. General game playing requires digital agents which are mapped with policies to play games in uncertain action space environments where the agents can learn and alter the state of the environments while learning from positive and negative reinforcements to maximize the concept of cumulative positive value rewards through trial and error. Policy mapping coalesces reinforcement learning agents from perceived environmental states to actions to be taken when in those states. This survey will explore general gameplay (GGP) with reinforcement learning evaluation. General mathematical and pragmatic approaches taken in the field of research. The current survey aims to examine the history, association, and recent advances in the field of general game playing with reinforcement learning and its subfields.

2 Origins of Game Play

To explain the significance of gameplay and its uses for reinforcement learning, I wish first to present some background information, context, and contrast about games. From ancient Mesopotamia, Sumerian, Egypt, and China, games like Senet, Checkers, backgammon, Meehan, and Go have played a critical role in the adaptation of our modern world[1]. These games and others like them have fundamentally challenged people for millennia to solve abstract problems, exercise logic, test memory, develop pattern recognition, make critical decisions both visually and analytically, and simulate real-world scenarios or environments[3]. When Chess began in the 6th Century India as a 64-square board game, called Chaturanga, research shows that it precisely modeled on the military forces of the day[4]. Chess has since then been used by many great leaders throughout history to symbolize warfare[4]. In 2006, during an excavation, the Bronze-Age city Shahr-i Sokhta city in Iran, archaeologists found what is claimed to be the first dice carved of stones dating back over 5000 years[5]. Mesoamerican ball games were created by the ancient Mayans and possibly invented by the Olmecs. The ball game goes back 3,500 years, making it the first organized game in the history of sports[6]. Today, games come in a wide variety of forms and are far more advanced. It was recently reported that 65% of families did online gaming during the COVID-19 pandemic in 2020 to interact with family and friends they could not meet in person[7]. Studies have shown that playing video games can increase sensitivity to contrasts, eye-to-hand coordination, and memory[3]. According to Reynaldo who reviewed 27 experimental and literature reviews on this subject [8]. The review showed that video games improve cognitive abilities. As a result of playing video games, subjects improve their cognitive skills such as perception, attention, and decision-making [8]. Games are part of human nature, culture, and history which have defined significance in our cognitive development and modern advances. With games, we can generate responsive, adaptive, and intelligent behavior or simulate real-world scenarios, training data models to actively learn tasks and achieve autonomy[9] Games play a significant role in our modern world and human development and have thus attracted a great deal of scientific interest. Learning takes place through trial and error in humans and animals alike. Behavioral responses are influenced by our human based reward mechanisms. Hence, games are ideally suited to integrating artificial intelligence and reinforcement learning through training models and systems to learn on their own based on the reinforcement of rewards and consequences for their actions[10].

3 History of Reinforcement Learning

Alan Turing's 1950 proposal to test if machines had intelligence, the field of artificial intelligence was born and has now advanced beyond science fiction and theory[11]. In recent years, deep reinforcement learning has shown remarkable success in various domains, including finance, medicine, healthcare, video games, robotics, and computer vision. By enhancing reinforcement learning frameworks and utilizing deep neural network representations. Deep reinforcement learning now powers some of the most prominent successes in artificial intelligence, from recommendation systems that learn from users' behavior to autonomous intelligent devices[12]. Modern reinforcement learning originated from a concept called optimal control derived from control theory in the late 1950s[13]. Optimal control is a nonlinear differential problem used to determine what control laws must be used to achieve an optimization criterion and determine the best control for the system[14]. In turn, these equations define a path that minimizes an error function value. Developed in the 1950s, Richard Bellman's work, specifically in dynamic programming, is at the core of optimal control. Dynamic programming is an optimization technique that seeks to solve a large problem by breaking it down into smaller, more manageable pieces. As well, it is regarded as the only feasible method for solving stochastic optimal control problems and is considered in general to be a reinforcement learning process[15]. Additionally, Richard Bellman contributed to what is commonly known as the Bellman equation in dynamic programming. This dynamic programming equation is based on a recursive formula that outlines a framework for determining the optimal expected reward at a state s which is used as the value function in reinforcement learning[16].

Bellman Equation

$$\boldsymbol{V(s,t) = \frac{max}{\alpha 0}\{\Delta R(s,t) + \beta V(T(s1,t))\}, s.t\, V(s,T) = D(X)}$$

Where $V(s,t)$ = the partial derivate of $Vw.r.t,$ the time variable $t, a \cdot b, V(s,t)$ = Bellman value function being the cost incurred from starting as an unknown scalar in state $s$at time t while controlling the system until time T,R, = the change in reward as per the previous and current state, V = the scalar cost rate function, D = the final utility state function, $s(t)$ = system state vector, $s(0)$= an assumed given, $u(t)\, for\, 0 \leq t \leq T$ The value function yielded by this equation is the minimum cost for a given dynamic system. Generally, dynamic programming is the only viable way or method to solve stochastic optimal control problems[15]. Problems involving stochastic optimal control involve an objective function that is uncertain when decisions are made to maximize or minimize it[17]. This equation computes the expected total reward of taking an action from a state$s$ in a Markov Decision Process(MDP) by breaking it down into the immediate reward and the total future expected reward. Markov Decision Process (MDP) models are models in which a decision maker, or agent, inhabits an environment whose state changes randomly in response to action choices made by the decision-maker[18]. In Ronald Howard's 1960 book, Dynamic Programming and Markov Processes (MDP) were introduced. It must be noted that dynamic programming algorithms for reinforcement learning (RL) require two strong assumptions. First, the environment model has to be known, and second, the state space has to be small enough so that the curse of dimensionality does not apply. Today, more modern algorithms such as Q-learning and R-learning are preferred methods of implementing reinforcement learning[19]. In the 1980s, the concept of learning by temporally successive predictions through a new concept of temporal difference learning (TD learning) was introduced by Richard S. Sutton and Andrew G Bartow[20]. The development of learning procedures specialized for prediction, i.e., for user experience with an incompletely known system to predict its future behavior. Rich Sutton, Ph.D. is currently Professor of Computer Science, iCORE chair at the University of Alberta, and a Distinguished Research Scientist at DeepMind[20]. His research in reinforcement learning led to the development of Deepminds AlphaGo Zero Project which in 2017 became the first artificial intelligence program to defeat 18-time world champion Lee Sedol in the game of Go[21]. This was a major milestone in human history since it was the first time a computer played a complex game without human guidance and won by itself. In addition, in 2014, complex models that used deep learning neural networks called deep reinforcement learning (DRL) were successfully introduced in application for facial recognition at Google call Deepface[22]. DRL is the use of deep learning algorithms implemented on neural networks and deep learning models with reinforcement learning, and it has achieved great popularity ever since the demonstration of the ability to play Atari games autonomously and win from converting raw image pixels into usable signal data as shown below in Table 2[23].

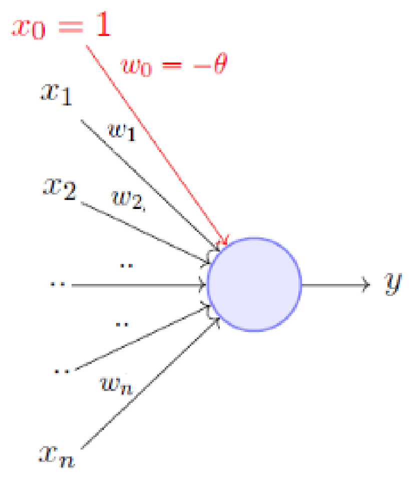
4 Reinforcement learning

Reinforcement learning is different from other machine learning paradigms like supervised and unsupervised machine learning. Essentially, reinforcement learning aims to maximize a variable under certain constraints. Reinforcement learning algorithms find an optimal path to achieve goals by using agents in actions spaces and situations where they are uncertain about the details. Reinforcement learning seeks to find a set of actions, mapped to a set of states, that produce the best possible outcome. In reinforcement learning, an agent uses data from its environment that it has collected through its observations to learn from experience. Initially, it does not know explicitly what to do until introduced to the game environment, then through trial and error learns what happens while updating the data set to make future predictions and decisions, keeps track of recent successful decisions, and makes them under similar circumstances in the future, and takes actions that alter the initial state[20], [24].
Fundamental functions and concepts that form the basis of reinforcement learning:

- ***Agent*****:** The computational entity that makes decisions and follows certain processes.
- ***Environment*****:** The digital environment in which an agent works, such as a game to win or task to accomplish.
- ***State*****:** Is what is produced by the environment and observed by the agent resulting in the current positions and moment of the agent in its environment.
- ***Action*****:** This is the next move that the agent chooses to take based on its decision process.
- ***Reward*****:** This is the feedback that describes how the agent should behave based on the input it received for making decisions in the environment.
- ***Policy*****:** This is a function π to map an agent's state to its actions. It is the definition and planning process of the agent
- ***Episode*****:** Simulation iteration at the end of which the system reaches a terminal state.
- ***Value function*****:** This is the future reward-based system (+- scalar value) that an agent will receive for taking actions based on future actions it could or decides to take. Value function will map the states to real numbers, with the value of one state representing the long-term outcome brought about by executing a particular policy.
- ***Model*****:** A state-action pair can be mapped to a probability distribution over states in the agent's view of the environment. However, not every reinforcement learning agent uses an environment-model.
- ***Function approximator*****:** This refers to the problem of constructing a function from training examples. An approximator can be either a decision tree, neural network, or nearest neighbor method.

4.1 Reinforcement Learning Principles:

Acting can reward or penalize the agent, as well as change the environment's state. Essentially, the agent's goal is to determine what action to take in each state to maximize its total rewards from the environment as a result of its actions. In the standard deep reinforcement-learning model, an agent is connected to its environment via perception which utilize backpropagation and action, as depicted in Figure 1 below. The perceptron takes an input, aggregates it (weighted sum), and returns 1 only if the aggregated sum is more than some threshold else returns 0 [25].



A more accepted convention,

$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

$$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$$

Figure 1: Perceptron

As seen in Figure 2 below, at each step of interaction the agent receives as input, i, with some indication of the current state, s, of the environment; the agent then chooses an action, a, to generate as output. The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar reinforcement signal,r. The agent's behavior B, should choose actions that tend to increase the long-run sum of values of the reinforcement signal. It can learn to do this over time by systematic trial and error, guided by a wide variety of algorithms that are the subject of later sections of this paper[26].

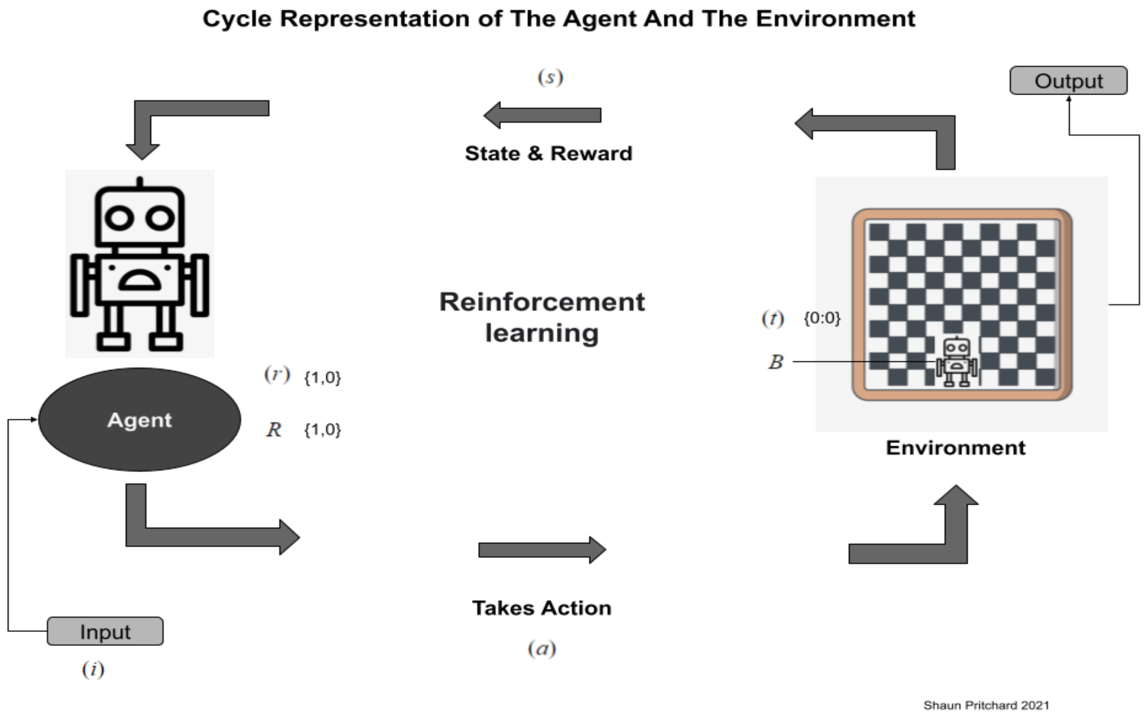


Figure 2: Reinforcement Learning Lifecycle

$i=$	Input signal(data)
$st \in S =$	The state, S is the state space of all posible states.
$at \in A =$	The action, $A$is the action space
$B =$	Agents behavior
$t = T =$	Time step terminal state in which these signals occurred
$r =$	Value of this state transition signal set of scalar {0,1} reinforcement signals
$rt = R(st, at, st+1)$ $i =$	The reward,R is the reward function
I	How the agent views the environment

5 Reinforcement Learning Environments

The reinforcement learning processes of encouraging or establishing systematic patterns of behavior, especially by reward or penalty are defined as follows:

- ➢ ***Positive:*** is reward based on the result of a specific behavior of the actor and is defined as an event. This increases the strength and frequency of the behavior and has a positive influence on the actions taken by the agent. Reinforcement of this type enables you to maximize performance and maintain change for a longer period. Over-optimization of state, however, may lead to poor results when too much reinforcement is applied.
- ➢ ***Negative:*** is the penalty-based or lack of reward-based negative reinforcement which refers to behavior that is reinforced because of a negative condition that should have stopped or been avoided. This enables you to determine the minimum level of behavior. This method has the disadvantage that it provides just enough to meet the minimum requirements.

5.1 Discrete and continuous environments

Generally, agents and agent policies interact with two types of environments as follows:

- ➢ ***Discrete environment*****:** Discrete environments have a discrete action space that consists of basic actions such as up, down, left, right.
- ➢ ***Continuous environment:*** Continuous environments are those where the action space of the environment is continuous. If, for instance, we were teaching an agent how to drive a car, then our action space would be continuous, with several continuous actions such as changing the speed of the car, adjusting the amount of rotation, and so on.

5.2 Episodic and non-episodic environments

- ➢ ***Episodic environments*** are nonsequential environments, an agent's present actions are not bound to affect their future actions.
- ➢ ***Non-episodic environments*** are sequential environments because an agent's current actions will affect future actions.

5.3 Single and multi-agent environments

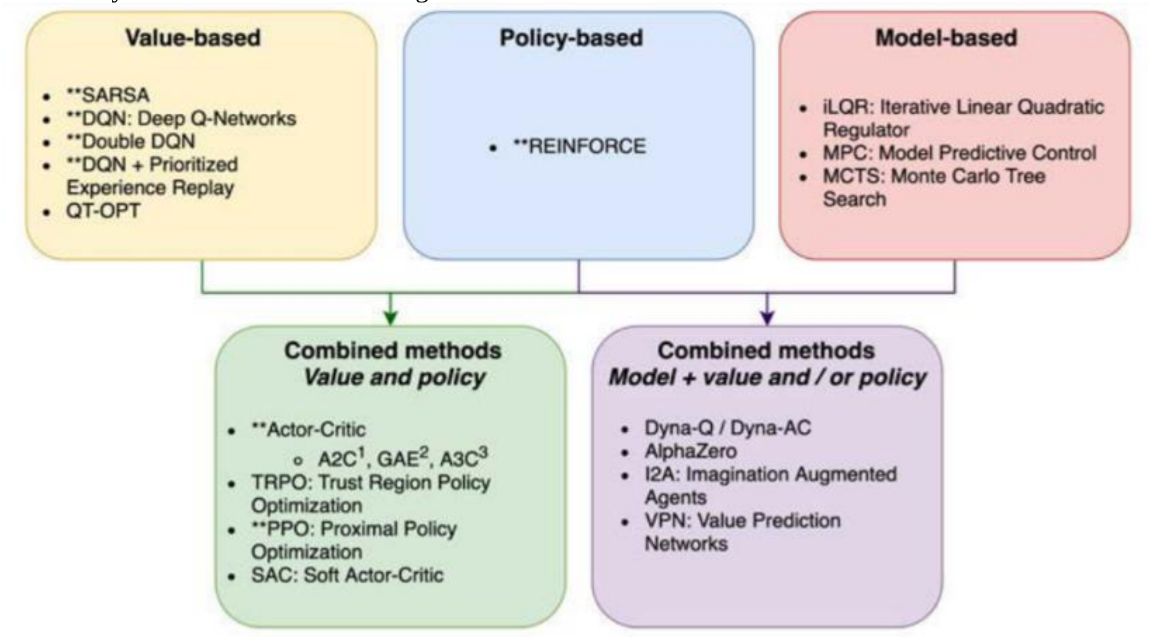
- ➢ ***Single agent environments*** are environments that exist when only one agent is involved
- ➢ ***Multi-agent environments*** are environments that exist when more than one agent is involved.

6 Reinforcement Learning Algorithms

Reinforcement learning, there are three main categories of algorithms. Depending on the end goal of model training, each approach employs different characteristics to achieve the best results as follows.

- ➢ ***Policy-based:*** develops a policy that thinks about the actions that must be taken in each state as so that they will lead to a maximum gain in the future. There are two types of policy-based methods as follows:
  - ○ ***Stochastic*****:** for every action produces a certain probability, which is represented by the following equation. Stochastic Policy :$n\{a\backslash s) = P\backslash A, = a\backslash S, = S]$.
  - ○ ***Deterministic*****:** involving the fact that, no matter what state a policy π is applied to, the same action will be produced.
- ➢ ***Policy-based*****:** learners are greedy gradient algorithms that focus on optimizing the policy function directly rather than trying to learn a value function that would yield information on the expected rewards in a given state. The agent is dependent on the policy.
- ➢ ***Off-policy*****:** learners learns the value of the optimal policy independently of the agent's actions.

- ➢ ***Value-based*:** consists of trying to maximize a value function$V(s)$. The agent uses this method to predict how the current state of policy π that will evolve over time and iterations.
- ➢ ***Model-Based*:** creates virtual models of each environment. The agent then learns how to perform in that environment.
- ➢ ***Combined Methods*:** Combined method or hybrid methods in which agents learn from multiple methods in the same environment. These methods can consist of value and policy-based methods or a mix of model, value, and policy. Figure 3 illustrates examples of algorithm categories and subsets based on the following methods [27].



Value-based
**SARSA
**DQN: Deep Q-Networks
**Double DQN
**DQN + Prioritized Experience Replay
QT-OPT
Policy-based
**REINFORCE
Model-based
iLQR: Iterative Linear Quadratic Regulator
MPC: Model Predictive Control
MCTS: Monte Carlo Tree Search
Combined methods Value and policy
**Actor-Critic
A2C1, GAE2, A3C3
TRPO: Trust Region Policy Optimization
**PPO: Proximal Policy Optimization
SAC: Soft Actor-Critic
Combined methods Model + value and / or policy
Dyna-Q / Dyna-AC
AlphaZero
I2A: Imagination Augmented Agents
VPN: Value Prediction Networks

Figure 3: Reinforcement Learning Algorithms

7 Reinforcement Learning Models

Reinforcement learning is based on only two models that can be tested over a set of data, with multiple approaches to providing an algorithm that explains how agents learn behavior and actions as a result of past actions and behaviors in the environment[28]. There are two variations of these two models. A *model-free* algorithm estimates the optimal policy without considering the dynamics of the environment (transition functions and reward functions). Alternatively, a *model-based* algorithm estimates the optimal policy by using the transition functions and reward function. The Markov Decision process model and Q-learning model are two reinforcement learning models that use model-free and model-based algorithms. There is a great deal of significance to each of these models, both in adversarial gameplay as well as in other reinforcement learning tasks. Here is a breakdown of each model[29]. The Markov Decision Process model provides a mathematical framework for modeling decisions in solving reinforcement learning problems[30]. MDP is used to model decisions that can have both probabilistic and deterministic rewards and punishments. MDP is a probabilistic model of a sequential decision problem, where states can be perceived exactly, and the current state and action selected to determine a probability distribution on future states. It is essential to first understand Markov properties and Markov chains to understand MDP[31]. ***The Markov property*** states that the future depends only on the present and not on the past. For any given time, the conditional distribution of future states of the process given present and past states depends only on the present state and not at all on the past states.

$$\boldsymbol{P(future \mid present, a0) \ = \ P \ [\ St+1 \mid St ...]}$$

$$\boldsymbol{Markov\ Property \ = \ st+1 \sim \ P\ (st + 1\ |(s0, a0), (s1, a1), ..., (st, at))}$$

Where;

$P\ is\ the\ NxN\ state\ transition\ probability,$
$an\ initial\ N\ x\ 1\ state\ vector\ = S = \ [S1\ , S2, S3, ...], t = \ time\ step$

Additionally, the next state st+1 is selected from a probability distribution P conditioned on the entire history at time step t. During an episode(iteration), an environment can transition between states s and s+1 based on the actions a and states' s. The outcome of applying an action to a state depends only on the current action and state (and not on preceding actions or states).

The Markov Chain is essentially when we evaluate multiple properties in a Markov process, a chain of these sequential state properties that strictly obey the Markov property. Where, $\boldsymbol{Pss' = P[St+1 = s' \mid st = s]}$

Markov Reward Process (MRP), extends the Markov chain to include a reward function. In other words, MRP is the Markov chain comprised of state representations, transition probabilities, as well as a reward function. Where, $Rs =$ $\boldsymbol{set\ of\ transition\ probabilities\ with\ immediate\ r}$ $\boldsymbol{E[Rt+1 \mid St = s]}$ ***Markov Decision Process (MDP),*** is basically an extension of the MRP with actions. MDP Incorporates actions where $A = \epsilon\, actions$into the Markov Reward Process. Where $MDP = m = (S, A, P, R, \boldsymbol{\gamma})$. According to the Markov property, the next state depends only on the current state and does not depend on the previous state. By using this model, the agent is able to make decisions only based on the current state, and not based on the past state. Acting in this way, the agent can determine the next state that will implement a new reward for those actions. An MDP has four components: states, transition probabilities, reward functions, and action functions. Where,

$S =$	Set of states in the environment.
$A =$	Set of actions the agent can perform in each state.
$P = P(s' \mid s, a) =$	The transition probability moving from a state s to state ss' performing an action.
$R =$	The reward functions.
$\boldsymbol{\gamma} = \in [\boldsymbol{0}, 1]$	The discount function decides on how future rewards are considered. (The more a future reward is calculated the less we take it into account)

Using MDP with policy distribution where, $\boldsymbol{policy} = \boldsymbol{\pi, \pi(a \mid s) = P[At = a \mid St = s]}$ The MDP allows the Agent to distribute actions based on the current state. As a result, reinforcing the agents' actions to gain rewards, which update the policy based on the agents' states[32]. To further optimize MDP and give the agent predictive capabilities, the value function which holds the long term value and return of a state or action is used where $V(s) = value\ function$. The Bellman Equation (as described above) gives a standard representation for the value function $V(s)$. By decomposing the value function into two components, it calculates the immediate reward and discounts the values of future states. Where,

$\boldsymbol{V(s) = Rs + \sum_{s' \epsilon S}^{\blacksquare} \blacksquare Pss'V(s)}$**.** We might use this process if we want our robot to find the shortest path to reach the red dot in Figure 4 below, avoiding obstacles that entail negative reward values, in the last possible number of moves. Negative values are represented by white squares, while positive values are represented by black squares. The agent must restart when he lands on lava. Using MDP, an agent receives a reward based on assessing the value function of each move it makes in the environment that alters the current state until it reaches its objective, which is the most efficient path[19].

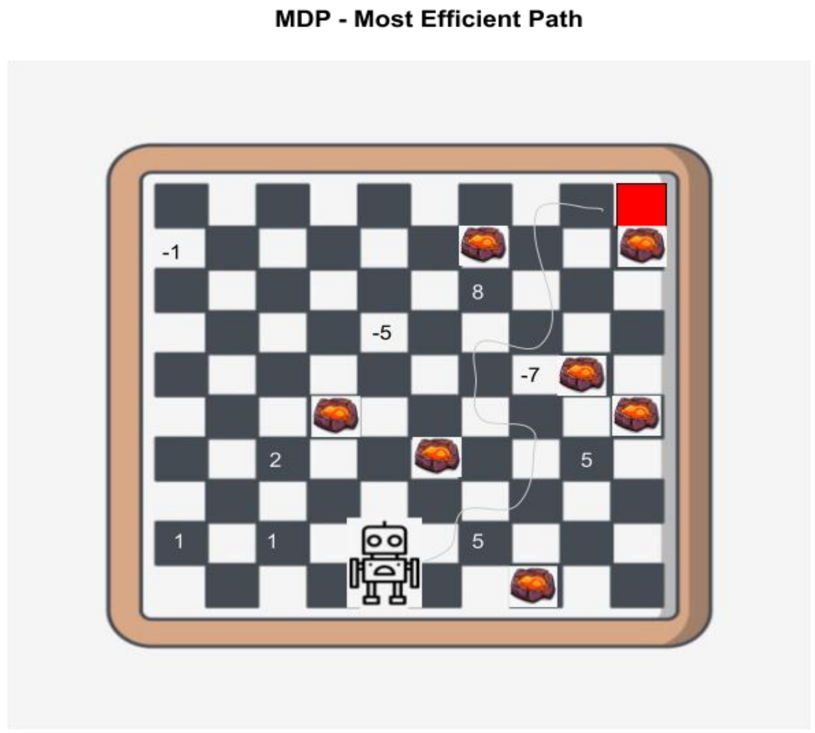


Figure 4: MDP most efficient path

***Q-learning* –** Q-learning is another model which is a value-based method of supplying information to inform the agent which action an agent should take by learning Q-value $Q(s, a)$as oppressed to the value function$V(s)$ in an environment. Q-learning Learns from actions that are outside the current policy. Q learning is a part of a family of model-free learning algorithms which learn a policy by looking at all of the possible actions and evaluating each of them. Q-learning uses the Q-function where: $\boldsymbol{Q(s, a) \ = E\pi\{Rt|st = s, at = a\} = E\pi\{\sum k = 0\infty\gamma krt + k + 1|st = s, at = a\}}$. Algorithms implemented in Q learning include Q learning, Deep Q learning, as well as Deep Deterministic Policy Gradients[12]. The expected Q-value is a cumulative discounted reward of some action in the current state and then following the optimal policy. The agent maintains a Q-table of $Q[s, a]$, where S is the set of states and A is the set of actions used to calculate the maximum future rewards. $Q[s, a]$represents its current estimate of Q*(s,a) where it learns each value of the Q-table using the Q-Learning algorithm. For general game play, Q-learning and deep Q-learning are the most widely used models. Q-learning is more efficient and widely used than other game play algorithms[33].

7.1 Policy and Value based algorithms

Game AI involves perception and decision-making in game environments. Today there are many different algorithms used for many different problems in gameplay using reinforcement learning. The following are 4 main categories of policy and value-based algorithms used in reinforcement learning as follows[27].

➢ ***REINFORCE algorithm,*** policy gradient algorithm where an agent learns a policy and uses this to act in an environment. The REINFORCE algorithm numerically estimates the policy gradient using Monte Carlo sampling. Monte Carlo sampling refers to any method that uses random sampling to generate data used to approximate a function. Monte Carlo methods are a class of methods used to learn value functions that evaluate the value of a state by averaging the total rewards received from many trials starting at that state.

- ***SARSA Algorithm***, The SARSA algorithm has two main elements, learning the Q-function with TD learning and a method for acting using the Q-value estimates.
- ***Deep Q-Networks algorithm***, DQN uses various methods containing Deep learning and Convolutional Neural Networks(CNN) with Q-learning to learn the optimal Q-function instead of the Q-function corresponding to the current policy. This makes DQN an off-policy algorithm in which the learned Q-function is independent of the experience-gathering policy.
- ***Improved DQN***, like the Deep Q-Network, is optimized to use two different networks to estimate the Q-value of the next state and prioritizes experience replay. This algorithm is most frequently used to play Atari games which I discuss in the next section.

7.2 Combined algorithms

Combined algorithms combine different policy and value-based methods for more enhanced algorithms. The following are noted characteristics of advanced combined algorithms.

- ***Advanced Actor-Critic (A2C & A3C)***, algorithms are composed of multiple components that must learn together. An actor, learning a parameterized policy, and a critic, learning a value function for evaluating state-action pairs.
- ***Proximal Policy(PPO)***, extend the Actor-Critic by modifying its policy loss, which improves the stability and sample efficiency of training.
- ***Parallelization,*** parallel synchronous and asynchronous reinforcement learning methods to handle the one-arm bandit problem and handle implementing multiple agents attempting to learn the value function at the same time.

All of REINFORCE, SARSA, A2C, and PPO are on-policy algorithms, while DQN and Double DQN + PER are off-policy algorithms. The SARSA, DQN, and Double DQN + PER algorithms learn to approximate the $Q\pi$ function. Thus, they can only be applied to discrete environments. REINFORCE is a policy-based algorithm which only learns a policy π. Both A2C and PPO are hybrid algorithms that learn a policy π and the $V \pi$ function. There are two types of action spaces: discrete and continuous, and REINFORCE, A2C, and PPO can all be used in both types of environments. Until now, we have discussed the computation and theory behind reinforcement learning methods used in general game play[23].

7.3 General Game Play (GGP) with Reinforcement Learning

The purpose of this section is to discuss the application of reinforcement learning techniques to various applications with general games. In addition, we will discuss the challenges, issues, and platforms used for gameplay with reinforcement learning. After Alpha Go's superhuman performance (computer outperforming humans) when beating the World champion at the game of Go in 2017, people assume that general gameplay (GGP) is implemented through notable reinforcement learning techniques such as Monte Carlo Q-learning. GGP is usually thought of for logic-based AI for board games like tic tac toe or chess. When in fact GGP far exceeds the mere limitations of simple board games into the realm of 2D and 3D asymmetric video games utilizing MDP and Q-learning deep neural networks[24]. Reinforcement learning in general gameplay implements AI agents which are programmed without any explicit knowledge of a particular game. Consequently, they must be written without explicit knowledge regarding any particular game, encouraging strategies that can be used across domains and general algorithms that produce agents that can plan and learn rather than simply using game-specific heuristics that humans have developed[33]. Reinforcement learning (RL) looks at agents which can learn to maximize the reward by taking actions and recording their effect[34]. As a result, Superhuman performance is the new benchmark to measure the distinction and performance of reinforcement learning algorithms on complex human-based board games and complex digital video games alike.

7.4 Supervised classification vs Reinforcement Learning scenario

Here is an illustration of what a supervised classification problem compared to reinforcement learning would look like in the real world. For example, if we were to solve the problem of training the AI to beat a 2D video game like Super Mario Brothers. We could use images or video frames from players who are best at beating the game as the labeled input, and the learners would use this data to produce Mario's movements and actions to beat the game as our output. The labels in this instance would be the directions to move and predefined actions of the best players. The downfall is this would only be efficient and perform well if we had sufficient data (100's of hours of video) to train and enough hardware and GPUs to handle the computational requirements to train this type of model. Also, we are training data on dynamic models on which consistent regenerative and emerging data will update the decision table adding to the unconfirmable amount of raw video footage for the agent to base its decisions on which is computationally expensive. This is why in General Game Play with reinforcement learners which use dynamic programming paradigms know nothing of the game before starting and use trial and error with rewards to solve the problem. Essentially, the agent has a specific number of actions it can perform while earning rewards that act like time-delayed labels that help determine future actions which lead to altering the state within the environment as a sequence of states actions and rewards per episode. The transition probability (Markov decision Process) of the state is determined by the initial preceding state and current action and current state but not by the states or actions before it. In turn, the agent plans for short and long-term rewards which base the model's performance on beating the parameters of the game. This is why reinforcement learning as opposed to supervised learning is a more accurate model for implementing general gameplay.

7.5 Challenges in gameplay with reinforcement learning

Implementing AI in gameplay offers several challenges. First, the action state space of the game is very large, especially in strategic games. With the rise of representation learning, deep neural networks can successfully model large-scale state spaces. Secondly, it is difficult to learn what policies to follow to make decisions in a dynamic, unknown environment[35]. For this problem, data-driven methods, such as supervised learning and reinforcement learning (RL), offer viable solutions. Most of the AI in video games is developed in a virtual environment. Transferring AI capabilities between games is a core challenge. Learning by reinforcement involves limited datasets in simulated environments, which limits the transfer of results to real-world situations. Furthermore, the high density of data required makes it extremely difficult to train agents in the action space so there is the challenge of dimensionality. Finally, the algorithm and model training performance are constrained to the computational resources i.e. (hardware, memory, network, GPU, and OS) system running the algorithm[36].

7.6 Applications of AI

Gaming and robotics are two of the most popular applications of reinforcement learning and AI. As discussed earlier, applications of AI in games hold value for real-world applications outside of training games to beat human players at games like Alpha Go in 2017[37]. Reinforcement learning is not restricted to these mediums though; it is now widely used in multiple industries such as finance and banking. These systems are helping in fraud detection by identifying risks in advance[38]. Deep learning has revolutionized healthcare and is now being used to make patient care easier and more accurate. Reinforcement learning is used by robots to conduct surgeries, use medical equipment, manage patient records, and detect disease early on[39].
Recent research published by Stanford shows reinforcement learning can now detect skin cancer faster than deterministic approaches from physicians[40]. With preemptive training of agents on different scenarios, military, business, and law enforcement can prevent crime and protect against dangerous attacks and exploits[41]. The application of reinforcement learning can be used for gathering intelligence, solving new problems, simulating environments, procedural content generation (agent builds game levels and environments)[42] developing autonomous vehicles, NLP, and many other applications. It involves using intelligent systems, starting with a virtual game environment, and then letting A. I algorithms find solutions to the problem.

7.7 Game Environments

Reinforcement learning can be implemented in a variety of virtual game environments. Virtual environments can be simulated in 2D or 3D digital space. It was noted earlier that the environment plays a major role in training the models and algorithms to solve the problem or win the game. In complex environments, the agent must interpret more states, thus interpreting more actions and values as well. Training models in 2D environments such as old 8-bit Atari games is a much simpler task than training models for a game like Call of Duty for example. The abstraction becomes more difficult as the frequency of the environment changes. Reinforcement learning can also be applied to more complex space environments, such as the augmented and virtual worlds. Several important limiting assumptions need to be inferred from reinforcement learning in the game space environment, such as whether it is a cooperative game, non-board game, asymmetric game, multiplayer, or simultaneous game. RL must be able to determine whether the game is zero-sum, symmetric between two players, turn-based and if it has a board or neural network architecture tailored for the attributes of the game in question[28].

7.8 General Game Play Methods

Since general gameplay reinforcement algorithms have no labels or instances to start with that describe the game before they play, models must be built that consider what learning agents to use. GGP addresses this by looking at programs that are only given the rules of the game at runtime. The lack of handcrafted heuristics means that performance should reflect the skill of the algorithm at that game, not the skill of the programmer. Typically, GGP implements the use of Upper Confidence Bound trees (UCT) with Monte Carlo Search Simulations or a neural network-based Monte Carlo Search Tree (MCTS), starting with an, initially empty, game tree, and runs several simulations until the optimal leaf is generated. Each one starts at the root of the game tree and proceeds down the tree using a variant on upper confidence bounds until a leaf node is reached the initial state and maximum score[26].

7.9 Deep reinforcement learning

As described above there are many different reinforcement models available for different problems implementing games[43]. Q-learning is an optimal solution that predicates the optimization model applied to Markov Decision Processes(MDP) that have proven to superhuman performance in generative and adversarial game playing. Deep learning convolutional neural networks(CNN) take tabular dynamic Q-learning and MDP approaches to the next level(no pun intended) by using multiple neural network layers to transform the pixels of any video game into models that can produce action and change the state of the environment be analyzed and inferred as shown in Figure 6 below[44].

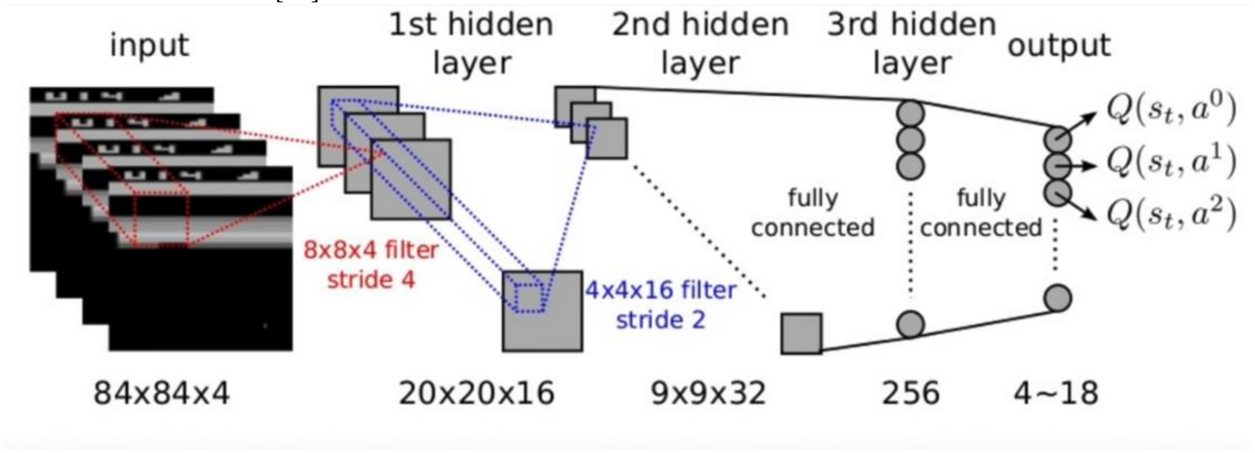
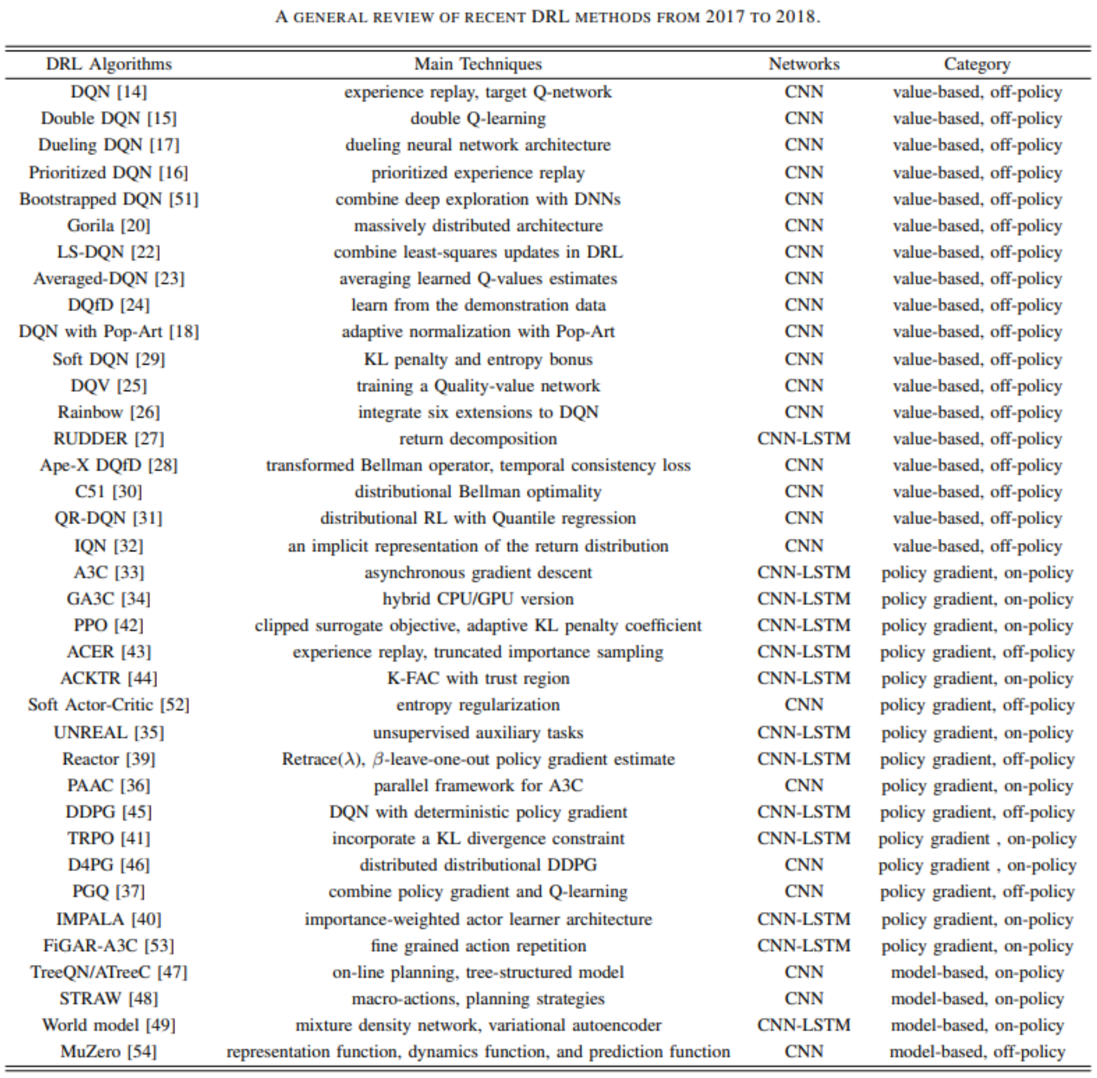


Figure 6: Deep CNN Q-Network (DQN)

General Video Game Playing is intended to design an agent to play multiple video games without human intervention. Using deep reinforcement learning many techniques implement the use of deep learning network models with reinforcement learning to achieve this goal as follows in Table 1 below:



A GENERAL REVIEW OF RECENT DRL METHODS FROM 2017 TO 2018.

DRL Algorithms	Main Techniques	Networks	Category
DQN [14]	experience replay, target Q-network	CNN	value-based, off-policy
Double DQN [15]	double Q-learning	CNN	value-based, off-policy
Dueling DQN [17]	dueling neural network architecture	CNN	value-based, off-policy
Prioritized DQN [16]	prioritized experience replay	CNN	value-based, off-policy
Bootstrapped DQN [51]	combine deep exploration with DNNs	CNN	value-based, off-policy
Gorila [20]	massively distributed architecture	CNN	value-based, off-policy
LS-DQN [22]	combine least-squares updates in DRL	CNN	value-based, off-policy
Averaged-DQN [23]	averaging learned Q-values estimates	CNN	value-based, off-policy
DQfD [24]	learn from the demonstration data	CNN	value-based, off-policy
DQN with Pop-Art [18]	adaptive normalization with Pop-Art	CNN	value-based, off-policy
Soft DQN [29]	KL penalty and entropy bonus	CNN	value-based, off-policy
DQV [25]	training a Quality-value network	CNN	value-based, off-policy
Rainbow [26]	integrate six extensions to DQN	CNN	value-based, off-policy
RUDDER [27]	return decomposition	CNN-LSTM	value-based, off-policy
Ape-X DQfD [28]	transformed Bellman operator, temporal consistency loss	CNN	value-based, off-policy
C51 [30]	distributional Bellman optimality	CNN	value-based, off-policy
QR-DQN [31]	distributional RL with Quantile regression	CNN	value-based, off-policy
IQN [32]	an implicit representation of the return distribution	CNN	value-based, off-policy
A3C [33]	asynchronous gradient descent	CNN-LSTM	policy gradient, on-policy
GA3C [34]	hybrid CPU/GPU version	CNN-LSTM	policy gradient, on-policy
PPO [42]	clipped surrogate objective, adaptive KL penalty coefficient	CNN-LSTM	policy gradient, on-policy
ACER [43]	experience replay, truncated importance sampling	CNN-LSTM	policy gradient, off-policy
ACKTR [44]	K-FAC with trust region	CNN-LSTM	policy gradient, on-policy
Soft Actor-Critic [52]	entropy regularization	CNN	policy gradient, off-policy
UNREAL [35]	unsupervised auxiliary tasks	CNN-LSTM	policy gradient, on-policy
Reactor [39]	Retrace(λ), β-leave-one-out policy gradient estimate	CNN-LSTM	policy gradient, off-policy
PAAC [36]	parallel framework for A3C	CNN	policy gradient, on-policy
DDPG [45]	DQN with deterministic policy gradient	CNN-LSTM	policy gradient, off-policy
TRPO [41]	incorporate a KL divergence constraint	CNN-LSTM	policy gradient , on-policy
D4PG [46]	distributed distributional DDPG	CNN	policy gradient , on-policy
PGQ [37]	combine policy gradient and Q-learning	CNN	policy gradient, off-policy
IMPALA [40]	importance-weighted actor learner architecture	CNN-LSTM	policy gradient, on-policy
FiGAR-A3C [53]	fine grained action repetition	CNN-LSTM	policy gradient, on-policy
TreeQN/ATreeC [47]	on-line planning, tree-structured model	CNN	model-based, on-policy
STRAW [48]	macro-actions, planning strategies	CNN	model-based, on-policy
World model [49]	mixture density network, variational autoencoder	CNN-LSTM	model-based, on-policy
MuZero [54]	representation function, dynamics function, and prediction function	CNN	model-based, off-policy

Table 1: DRL methods

Published in 2015, was a paper titled Human-Level Control Through Deep Reinforcement Learning, in which researchers developed an artificial agent with a deep Q-network that is capable of learning policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning[23]. Atari 2600 classic games were used to test this agent's capabilities. Using the same algorithm, network architecture, and hyperparameters, the deep Q-network agent surpasses the performance of all previous algorithms and achieves a level comparable to the performance of a professional human games tester across a set of 49 games. It provided a bridge between high-dimensional sensory inputs and actions, resulting in the creation of the first artificial agent that can learn to perform well in a variety of challenging tasks. Table 2 shows the superhuman performance benchmarks applied to the 46 Atari games from the same agent and learner[23].

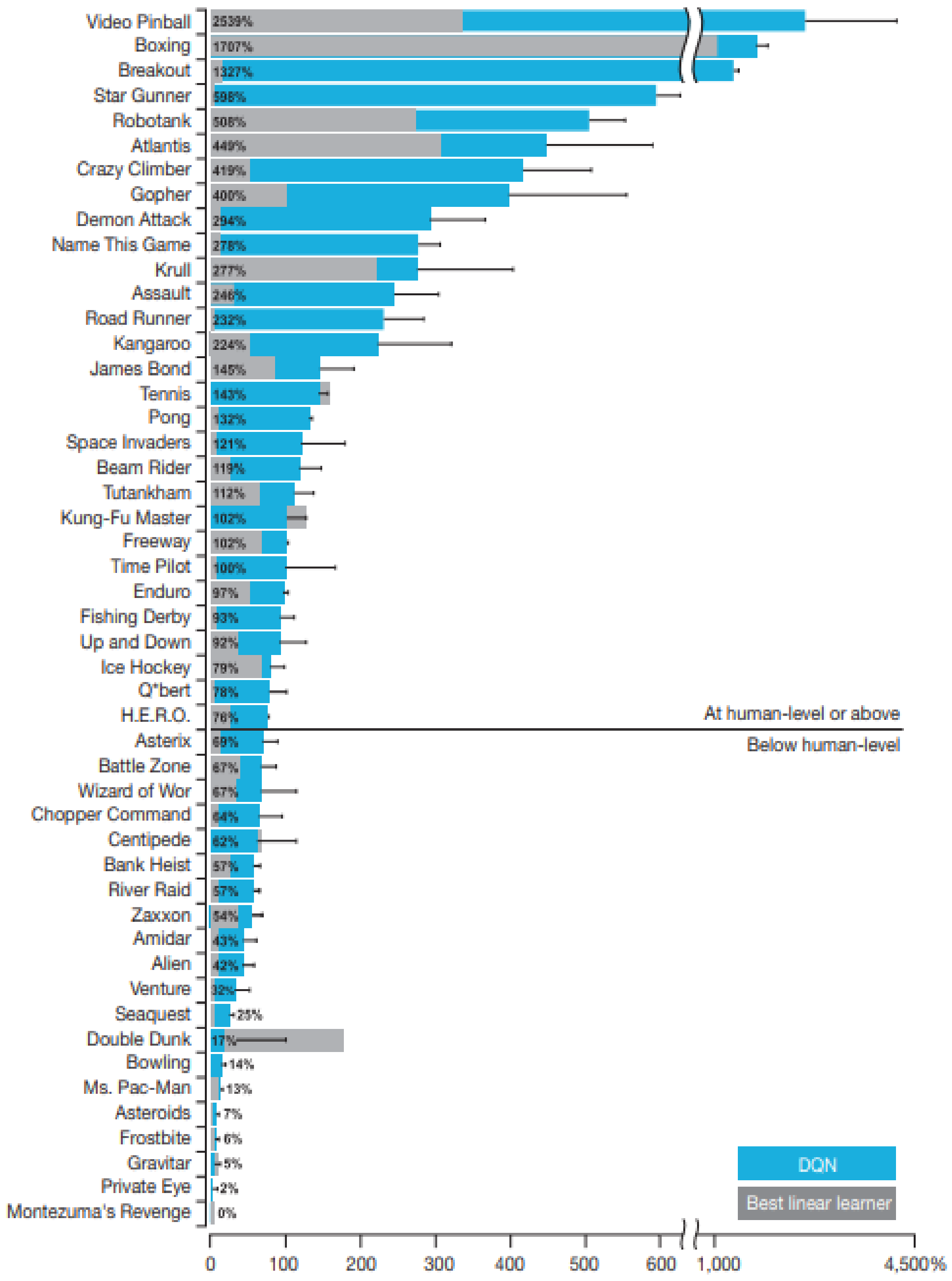


Table 2 DQN game performance

7.10 Research Platforms and Competitions

With openly available resources and tools, the advancement of gameplay with reinforcement learning would not have been so far. In this part of the survey, the focus is on game research platforms and competitions, and progress from 2D to 3D video games, from a single agent to multi-agent, non-model to model, implementing various general game playing techniques and deep learning models with Q-learning and reinforcement learning. A practitioner has the ability today to implement multiple platforms to test learning algorithms and models for research or real-world applications. There are many General Platforms(GP) available today that are open source and developed by some of the biggest names in tech. Here is a curated list of the most notable platforms and competitions used for reinforcement learning in general gameplay.

7.11 Reinforcement Learning Environments and Platforms

- ***Google Dopamine*** is a research framework for fast prototyping of reinforcement learning algorithms[45].
- ***OpenAI Gym*** integrates a collection of reinforcement learning tasks into a platform called Gym[46].
- ***OpenAI Universe*** is a platform for measuring and training agents' general intelligence across a large supply of games[47].
- ***Arcade Learning Environment*** (ALE) is the pioneer evaluation platform for DRL algorithms, which provides an interface to plenty of Atari 2600 games. ALE presents both game images and signals, such as player scores, which makes it a suitable testbed[48].
- ***Gym Retro*** is a wrapper for video game emulator with a unified interface as Gym, and makes Gym easy to be extended with a large collection of video games, not only Atari but also NEC, Nintendo, and Sega, for RL research[49].
- ***DeepMind*** Lab is a platform that helps in general artificial intelligence research by providing 3-D reinforcement learning environments and agents. Deepmind lab has first-person perspective learning environment, and provides multiple complicated tasks in partially observed, large-scale, and visually diverse worlds[50].
- Tensor Trade is an open-source python framework that uses deep reinforcement learning for training, evaluation, and deployment of trading strategies[51].
- ***VIZDoom*** lets you create an RL agent to play the well-known first-person player game called Doom[52].
- OpenSpiel is a diverse set of environments and algorithms that focuses on research in the implementation of reinforcement learning with games that involve search and planning[53].
- ***Ns3*** is a Network Simulator that helps in the understanding of networking protocols and technologies used for communication purposes[54].
- ***RecoGym*** is a reinforcement learning platform built on top of the OpenAI Gym that helps you create recommendation systems primarily for advertising for e-commerce using traffic patterns[55].
- **OpenSim** is another innovative reinforcement learning environment that can be used for designing AI-powered controllers to achieve various kinds of locomotion tasks[56].
- **TextWorld**, an open-source engine built by Microsoft, is beneficial in generating and simulating text games[57].
- **AirSim** combines the powers of reinforcement learning, deep learning, and computer vision for building algorithms that are used for autonomous vehicles[58].
- Project Malmo is an OpenAI gym like platform built over Minecraft, aimed for boosting research in Artificial Intelligence[59].
- ***AI Safety Grid worlds*** is a suite of environments used for depicting safety features of intelligent agents. Its environments are based on Markov Decision Processes and consist of 10×10 that can be customized as per the required simulation[60].
- ***AWS DeepRacer*** is a cloud-based 3D racing environment for reinforcement learning where you have to train an actual fully autonomous 1/18th scale racer car that has to be purchased separately. There are virtual and physical leagues that are officially hosted by AWS for DeepRacer for competition[61].
- ***DeepMind Control*** Suite is another reinforcement learning environment by DeepMind, that consists of physics-based simulations for RL agents[62].
- ***StarCraft II Learning Environment*** is a Python component of DeepMind, used for python based RL environment development[63].
- ***ReAgent*** is Facebook's end-to-end reinforcement learning platform that is open-source and helps in building products and services for large-scale[64].

7.12 Competitions

- ***The OpenAI Retro contest*** aims at exploring the development of DRL that can generalize from previous experience[65].
- ***The General Video Game AI*** (GVGAI) [66] competition is proposed to provide a easy-to-use and open-source platform for evaluating AI methods, including DRL[66].
- ***AI Arena SC2***, The SC2 AI Arena ladder provides an environment where Scripted and Deep Learning AIs fight in StarCraft 2[67].

- ➢ ***MineRL*** is a Minecraft based reinforcement learning competition concerned with the development of sample efficient deep RL algorithms[68].
- ➢ ***NetHack***, is a competition at NeurIPS 2021 in which teams will compete to build the best agents to play the game of NetHack[69].

8 Conclusion

I have presented an evaluation of mathematical arguments, historical context, applicable theory, environments, and evaluation of General Game Play with Reinforcement Learning (GGP-RL). Algorithms based on RL can be used to solve dynamic control and decision-making problems. As RL algorithms deal with continuous states and input variables, they rely on function approximations to represent the value function and policy mappings to reach their end goal. General game-playing AI with deep reinforcement learning is a challenging and promising direction for A.I. Recent advancements in this field have inspired the pursuit of artificial general intelligence (AGI) and solving real-world problems across a wide range of industries. A hypothetical intelligent agent with artificial general intelligence would be able to understand or learn any intellectual task that a human can perform. Since there are readily available and open source tools, resources, and ongoing research being done in the RL field, this notion would not be too far-fetched. To create a high-level AI in complex environments, we will need to explore more efficient and robust DRL techniques. With general gameplay and reinforcement learning, artificial intelligence has made many great signs of progress in the last decade or so. As far as research opportunities and advancements in this field are concerned, we are at an opportune time. In comparison, general gaming applications are robust. The goal of my future research will be to investigate and apply new variants of reinforcement learning, such as combined models, Deep Q-learning (DQN) with A3C, genetic programming methods, general artificial intelligence (GAI), Computer vision, and autonomy. The field of reinforcement learning offers a robust niche of educational resources that I find appealing and one of the most promising areas of AI.

9 References

[1] J. P. Blomster and V. E. Salazar Chávez, “Origins of the Mesoamerican ballgame: Earliest ballcourt from the highlands found at Etlatongo, Oaxaca, Mexico,” *Science Advances*, vol. 6, no. 11, 2020, doi: 10.1126/sciadv.aay6964.

[2] Payvand, “World’s Oldest Backgammon Discovered In Burnt City,” *Payvand*, Apr. 04, 2004.

[3] C. Reynaldo, R. Christian, H. Hosea, and A. A. S. Gunawan, “Using Video Games to Improve Capabilities in Decision Making and Cognitive Skill: a Literature Review,” *Procedia Computer Science*, vol. 179, no. 1, pp. 211–221, 2015, doi: 10.1016/j.procs.2020.12.027.

[4] BBC, “Has chess got anything to do with war,” *BBC*, May 03, 2015.

[5] Tasmin News Agency, “Iran’s Shahr-i Sokhta, A Treasure for Archeologists,” *Tasmin News Agency*, Sep. 26, 2016.

[6] G. Blümchen, “The Maya Ball Game,” *Cardiology*, vol. 113, no. 4, pp. 231–235, 2009, doi: 10.1159/000203640.

[7] BBC, “How online gaming has become a social lifeline,” *BBC*, Apr. 16, 2020.

[8] C. Reynaldo, R. Christian, H. Hosea, and A. A. S. Gunawan, “Using Video Games to Improve Capabilities in Decision Making and Cognitive Skill: A Literature Review,” *Procedia Computer Science*, vol. 179, no. 1, pp. 211–221, 2021, doi: 10.1016/j.procs.2020.12.027.

[9] E. Boyle, T. M. Connolly, and T. Hainey, “The role of psychology in understanding the impact of computer games,” *Entertainment Computing*, vol. 2, no. 2, pp. 69–74, 2011, doi: 10.1016/j.entcom.2010.12.002.

[10] E. Boyle, T. M. Connolly, and T. Hainey, “The role of psychology in understanding the impact of computer games,” *Entertainment Computing*,

vol. 2, no. 2, pp. 69–74, 2011, doi: 10.1016/j.entcom.2010.12.002.

[11] Allen Turing, “Computing Machinery and Intelligence,” *A Quarterly Review Of Psychology And Philosophy*, vol. 236, Oct. 1950.

[12] M. Wiering, *Reinforcement Learning: State-Of-The-Art (Adaptation, Learning, And Optimization)*, 1st ed. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-27645-3.

[13] L. Evans, “An Introduction to Mathematical Optimal Control Theory Version 0.2,” Department of Mathematics University of California, Berkeley, 1983. [Online]. Available: https://math.berkeley.edu/~evans/control.course.pdf

[14] E. Todorov, “Optimal Control Theory,” MIT Press, 2006. [Online]. Available: https://homes.cs.washington.edu/~todorov/courses/amath579/Todorov_chapter.pdf

[15] M. I. Gomoyunov, “Dynamic Programming Principle and Hamilton--Jacobi--Bellman Equations for Fractional-Order Systems,” *SIAM Journal on Control and Optimization*, vol. 58, no. 6, pp. 3185–3211, Jan. 2020, doi: 10.1137/19M1279368.

[16] X. Han, “A Mathematical Introduction to Reinforcement Learning,” 2018. [Online]. Available: https://cims.nyu.edu/~donev/Teaching/WrittenOral/Projects/XintianHan-WrittenAndOral.pdf

[17] P. Marbach and J. N. Tsitsiklis, “Simulation-based optimization of Markov reward processes,” *IEEE Transactions on Automatic Control*, vol. 46, no. 2, pp. 191–209, 2001, doi: 10.1109/9.905687.

[18] M. L. Littman, “Markov Decision Processes,” *International Encyclopedia of the Social & Behavioral Sciences*, vol. 1, no. 0, pp. 9240–9242, 2001, doi: 10.1016/b0-08-043076-7/00614-8.

[19] Ronald Howard, *Dynamic Programming and Markov Processes*, 1st ed., vol. 1. Massachusetts: John Wiley & Sons, Inc., New York : London, 1960.

[20] Richard S Sutton and Andrew G Bartow, *Reinforcement Learning An Introduction*, 2nd ed., vol. 2. Cambridge, Massachusetts: The MIT Press, 2015.

[21] David Silver and Aja Huang, “Mastering the game of Go with deep neural networks and tree search,” *Nature* , vol. 1, no. 529, pp. 484–489, Nov. 2015.

[22] Y. Taigman, M. Y. Marc’, A. Ranzato, and L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification”.

[23] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[24] D. Dwibedi, “Playing Games with Deep Reinforcement Learning,” 2018. [Online]. Available: https://debidatta.github.io/assets/10701_final.pdf

[25] X. Han, “A Mathematical Introduction to Reinforcement Learning,” 2018. [Online]. Available: https://cims.nyu.edu/~donev/Teaching/WrittenOral/Projects/XintianHan-WrittenAndOral.pdf

[26] A. Goldwaser and M. Thielscher, “Deep Reinforcement Learning for General Game Playing,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, pp. 1701–1708, 2020, doi: 10.1609/aaai.v34i02.5533.

[27] Laura Graesser and Wah Loon Keng, *Foundations of Deep Reinforcement Learning*, 2nd ed., vol. 1. New York: Addison Wesley, 2019.

[28] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. Perez-Liebana, “Deep Reinforcement Learning for General Video Game AI,” *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, vol. 1, no. 1, pp. 1–10, 2018, doi: 10.1109/cig.2018.8490422.

[29] A. Goldwaser and M. Thielscher, “Deep Reinforcement Learning for General Game Playing,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, pp. 1701–1708, 2020, doi: 10.1609/aaai.v34i02.5533.

[30] P. Marbach and J. N. Tsitsiklis, “Simulation-based optimization of Markov reward processes,” *IEEE Transactions on Automatic Control*, vol. 46, no. 2, pp. 191–209, 2001, doi: 10.1109/9.905687.

[31] A. Gouberman and M. Siegle, “Markov Reward Models and Markov Decision Processes in Discrete and Continuous Time: Performance Evaluation and Optimization,” 2012. [Online]. Available: https://www.unibw.de/technische-informatik/mitarbeiter/wissenschaftliche-mitarbeiter-innen/gouberman/rocks2014_mrms_and_mdps_preprint.pdf

[32] G. Ciardo, R. A. Marie, B. Sericola, and K. S. Trivedi, “Performability analysis using semi-

Markov reward processes," *IEEE Transactions on Computers*, vol. 39, no. 10, pp. 1251–1264, 1990, doi: 10.1109/12.59855.

[33] A. Goldwaser and M. Thielscher, "Deep Reinforcement Learning for General Game Playing," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, pp. 1701–1708, 2020, doi: 10.1609/aaai.v34i02.5533.

[34] A. Goldwaser and M. Thielscher, "Deep Reinforcement Learning for General Game Playing," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, pp. 1701–1708, 2020, doi: 10.1609/aaai.v34i02.5533.

[35] R. Fjelland, "Why general artificial intelligence will not be realized," *Humanities and Social Sciences Communications*, vol. 7, no. 1, 2020, doi: 10.1057/s41599-020-0494-4.

[36] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of Real-World Reinforcement Learning," 2019. [Online]. Available: https://arxiv.org/pdf/1904.12901.pdf

[37] M. Deepmind *et al.*, "Rainbow: Combining Improvements in Deep Reinforcement Learning," 2017. [Online]. Available: https://arxiv.org/pdf/1710.02298.pdf

[38] M. Deepmind *et al.*, "Rainbow: Combining Improvements in Deep Reinforcement Learning," 2017. [Online]. Available: https://arxiv.org/pdf/1710.02298.pdf

[39] "Frontiers | Deep Learning and its Application for Healthcare Delivery in Low and Middle Income Countries | Artificial Intelligence." https://www.frontiersin.org/articles/10.3389/frai.2021.553987/full (accessed Nov. 29, 2021).

[40] "Dermatologist-level classification of skin cancer with deep neural networks." https://cs.stanford.edu/people/esteva/nature/ (accessed Nov. 29, 2021).

[41] P. Contardo, P. Sernani, N. Falcionelli, and A. F. Dragoni, "Deep learning for law enforcement: a survey about three application domains," 2021, Accessed: Nov. 29, 2021. [Online]. Available: http://ceur-ws.org

[42] A. Khalifa, P. Bontrager, S. Earle, and J. Togelius, "PCGRL: Procedural Content Generation via Reinforcement Learning," 2020. [Online]. Available: https://arxiv.org/pdf/2001.09212.pdf

[43] D. Dwibedi, "Playing Games with Deep Reinforcement Learning," 2018. [Online]. Available: https://debidatta.github.io/assets/10701_final.pdf

[44] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. Perez-Liebana, "Deep Reinforcement Learning for General Video Game AI," *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, vol. 1, no. 1, pp. 1–10, 2018, doi: 10.1109/cig.2018.8490422.

[45] Google, "Google Dopamine:" https://github.com/google/dopamine (accessed Nov. 29, 2021).

[46] OpenAI, "OpenAI Gym." https://gym.openai.com/ (accessed Nov. 29, 2021).

[47] OpenAI, "OpenAI Universe." https://openai.com/blog/universe/ (accessed Nov. 29, 2021).

[48] Jesse Farebro, "Arcade-Learning-Environment," *GitHub*, 2021. https://github.com/mgbellemare/Arcade-Learning-Environment (accessed Nov. 29, 2021).

[49] OpenAI, "Gym Retro," *OpenAI*, May 25, 2018. https://openai.com/blog/gym-retro/ (accessed Nov. 29, 2021).

[50] Google, "DeepMind Lab | DeepMind." https://deepmind.com/research/open-source/deepmind-lab (accessed Nov. 29, 2021).

[51] Open Source, "TensorTrade." https://www.tensortrade.org/en/latest/ (accessed Nov. 29, 2021).

[52] Poznan University of Technology, "ViZDoom," *Poznan University of Technology*, 2021. http://vizdoom.cs.put.edu.pl/ (accessed Nov. 29, 2021).

[53] DeepMind, "OpenSpiel A Framework For Reinforcement Learning in Games." https://deepmind.com/research/open-source/openspiel (accessed Nov. 29, 2021).

[54] nsnam, "NS3," 2011. https://www.nsnam.org/ (accessed Nov. 29, 2021).

[55] RecoGym, "RecoGym." https://github.com/criteo-research/reco-gym (accessed Nov. 29, 2021).

[56] SimTK, "SimTK: OpenSim: Project Home." https://simtk.org/projects/opensim (accessed Nov. 29, 2021).

[57] Microsoft Research, "TextWorld." https://www.microsoft.com/en-us/research/project/textworld/ (accessed Nov. 29, 2021).

[58] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," *Springer Proceedings in Advanced Robotics*, vol. 5, pp. 621–635, 2018, doi: 10.1007/978-3-319-67361-5_40.

[59] Microsoft Research, “Project Malmo.” https://www.microsoft.com/en-us/research/project/project-malmo/ (accessed Nov. 29, 2021).

[60] Google Deepmind, “AI Safety Gridworlds | DeepMind.” https://deepmind.com/research/open-source/ai-safety-gridworlds (accessed Nov. 29, 2021).

[61] Amazon, “AWS DeepRacer - the fastest way to get rolling with machine learning.” https://aws.amazon.com/deepracer/ (accessed Nov. 29, 2021).

[62] Deepmind, “DeepMind Control Suite Dataset | Papers With Code.” https://paperswithcode.com/dataset/deepmind-control-suite (accessed Nov. 29, 2021).

[63] Deepmind, “StarCraft II Learning Environment.” https://github.com/deepmind/pysc2 (accessed Nov. 29, 2021).

[64] ReAgent, “ReAgent: Applied Reinforcement Learning Platform.” https://reagent.ai/ (accessed Nov. 29, 2021).

[65] Open AI, “OpenAI Retro Contest.” https://contest.openai.com/2018-1/ (accessed Nov. 29, 2021).

[66] DeepMind, “The GVG-AI Competition.” http://www.gvgai.net/ (accessed Nov. 29, 2021).

[67] SC2, “SC2 AI Arena.” https://aiarena.net/ (accessed Nov. 29, 2021).

[68] Mincraft, “MineRL: Towards AI in Minecraft.” https://minerl.io/ (accessed Nov. 29, 2021).

[69] NeurIPS, “NetHack .” https://nips.cc/Conferences/2021/ (accessed Nov. 29, 2021).