machine learning & knowledge extraction

Review

Recent Advances in Deep Reinforcement Learning Applications for Solving Partially Observable Markov Decision Processes (POMDP) Problems: Part 1—Fundamentals and Applications in Games, Robotics and Natural Language Processing

Xuanchen Xiang * and Simon Foo *

Department of Electrical and Computer Engineering, FAMU-FSU College of Engineering,
Tallahassee, FL 32310, USA

* Correspondence: XX16@MY.FSU.EDU (X.X.); foo@eng.famu.fsu.edu (S.F.)

Abstract: The first part of a two-part series of papers provides a survey on recent advances in Deep Reinforcement Learning (DRL) applications for solving partially observable Markov decision processes (POMDP) problems. Reinforcement Learning (RL) is an approach to simulate the human's natural learning process, whose key is to let the agent learn by interacting with the stochastic environment. The fact that the agent has limited access to the information of the environment enables AI to be applied efficiently in most fields that require self-learning. Although efficient algorithms are being widely used, it seems essential to have an organized investigation—we can make good comparisons and choose the best structures or algorithms when applying DRL in various applications. In this overview, we introduce Markov Decision Processes (MDP) problems and Reinforcement Learning and applications of DRL for solving POMDP problems in games, robotics, and natural language processing. A follow-up paper will cover applications in transportation, communications and networking, and industries.

Keywords: reinforcement learning; deep reinforcement learning; Markov decision process; partially observable Markov decision process

Citation: Xiang, X.; Foo, S. Recent Advances in Deep Reinforcement Learning Applications for Solving Partially Observable Markov Decision Processes (POMDP) Problems: Part 1—Fundamentals and Applications in Games, Robotics and Natural Language Processing. *Mach. Learn. Knowl. Extr.* **2021**, 3, 554–581. https://doi.org/10.3390/make3030029

Academic Editor: Ausif Mahmood

Received: 8 June 2021
Accepted: 25 June 2021
Published: 15 July 2021

1. Introduction

1.1. Markov Decision Processes

Markov Decision Process is a general mathematical problem representing an optimal path of sequential decisions in an uncertain environment. At each step of the sequence, the agent (actor) decides on an action and move to the next state. According to the current state, some rewards are available to get either positive gains or negative costs. Another characteristic of MDP is the uncertainty in the consequential state regarding the action taken.

As mentioned above, MDPs involve four sets of components: States S, Actions A, Transition probabilities $P(s'|s,a)$ and Rewards $R(s,a)$. Figure 1 can represent the MDP problems.

Solutions of MDPs are policies. A policy is a strategy and a rule specifying what action to execute in every possible state, denoted as $\pi(s)$.

Due to the uncertainty, following a policy yields a random path: s_0, $a_1 r_1 s_1$, $a_2 r_2 s_2$, ... Solving MDPs is to search for policies that maximize the rewards obtained by the agents. In this report, we care more about the recent rewards, and the future rewards are discounted by a factor γ ($0 < \gamma < 1$), called γ-discounted criterion. The sum of rewards, from state s, on this random path is the utility of the policy $U_\pi(s)$:

$$U_\pi(s) = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots + \gamma^t r_t + \ldots = \sum_{t=0}^{\infty} \gamma^t R(s_t) \tag{1}$$

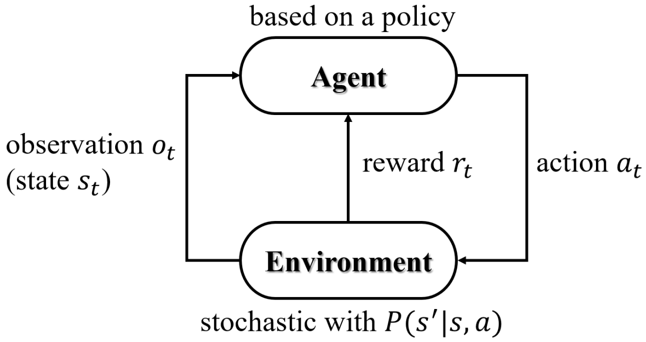

Figure 1. Markov Decision Process Model.

The expected utility following policy π from state s is the state value function $V_\pi(s)$ of the policy, which is not random:

$$V_\pi(s) = E[U^\pi(s)] = E[\sum_{t=0}^{\infty} \gamma^t R(s_t)] \tag{2}$$

State-action value function $Q_\pi(s,a)$, also called Q-value, of a policy is the expected utility of taking action a from state s, then following policy π:

$$Q_\pi(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_\pi(s')] \tag{3}$$

When it is not in the end state, the value is equal to the Q-value of the policy. This yields the Bellman Equation:

$$V_\pi(s) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_\pi(s')] \tag{4}$$

Bellman Equation is a recursive equation, as shown. Therefore, to find the optimal policy, Value Iteration or Policy Iteration can be utilized. Value Iteration is to get directly at the maximum expected utility. $V_{opt}(s)$ is assigned as the optimal value attained by any policy, and $Q_{opt}(s)$ is the optimal Q-value of any policy. When it is not in the end state:

$$V_{opt}(s) = \max_{a \in Actions} Q_{opt}(s,a) \tag{5}$$

Below is referred to as the Bellman Optimality Equation, solving which conducts the value of the optimal policy:

$$V_{opt}^{(t)}(s) = \max_{a \in Actions} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_{opt}^{(t-1)}(s')] \tag{6}$$

Policy Iteration randomly initializes the policy π and then solves the Bellman Equation to get $V_\pi(s)$. Then update the policy according to the greedy policy until it converges:

$$\pi_{opt}(s) = \arg\max_a Q_{opt}(s,a) \tag{7}$$

More algorithms will be discussed in Section 1.3.

1.2. Partially Observable Markov Decision Processes

In Completely Observable MDPs, a series of policies are given, while in reality, state measurements are often partially observable. The problems in such settings formulate Partially Observable Markov Decision Processes (POMDPs). POMDP is a generalization of MDP. It is a more complex scenario with the agent has no access to the underlying states. Instead, it must maintain a probability distribution over a set of states based on observations and observation probabilities.

There are six essential components, S, A, T, R, O and Z, in a POMDP. Apart from parameters similar to those in MDPs, O is a set of observations, T is a set of state transition probabilities, and Z is a set of observation probabilities. At each time step, the agent is in some state s and takes an action a, then the environment transits to state s' with probability $T(s'|s, a)$. Based on the new state s' and the action a, the agent receives an observation $o \in O$ with probability $Z(o|s', a)$. Furthermore, a reward $R(s, a)$ is obtained.

However, in POMDP, the agent does not know what the current state is. The solution is to transfer a POMDP to a belief MDP. The information about states is available as belief states, and b is a probability distribution over states:

$$b = [b(s^1), b(s^2), b(s^3), ...]$$

where $b(s^i)$ indicates the probability that $s^i \in S$ is the current state. By adding the concept of belief states, POMDPs can be regarded as traditional MDPs, with belief states as complete observable states.

At each time step, the agent is in a belief state $b(s)$, according to some policy π, the agent takes an action a and transits to the new physical state s' and observes o, given a reward $r(b, a)$. The reward is not observed but can be computed from the original POMDP:

$$r(b, a) = \sum_{s \in S} b(s) R(s, a) \tag{8}$$

The new belief state vector is b', in which an element is $b'(s')$. $b'(s')$ represents the probability of being in state s' after b, a and o, calculated as:

$$\begin{aligned} b'(s') &= Pr(s'|b, a, o) \\ &= \sum_s Pr(s|b, a, o) Pr(s'|b, a, o, s) \\ &= \sum_s b(s) Pr(s'|a, o, s) \\ &= \sum_s b(s) \frac{Pr(o|s', a, s) Pr(s'|a, s)}{Pr(o|a, s)} \\ &= \frac{\sum_s b(s) Z(o|s', a) T(s, a, s')}{\sum_s \sum_{s'} b(s) Z(o|s', a) T(s, a, s')} \end{aligned} \tag{9}$$

That is, for each possible new state s', $b'(s')$ can be calculated with known components: the current belief state, the observation probability, and the original transition probability.

The Value Iteration in POMDPs are similar in traditional MDPs, replacing states with belief states, the recurrence equation becomes:

$$V_{opt}^{(t)}(s) = \max_{a \in Actions} [r(b, a) + \gamma \sum_o Pr(o|b, a) V_{opt}^{(t-1)}(b')] \tag{10}$$

1.3. Reinforcement Learning

Machine Learning has three categories: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. RL is a framework used to solve problems that can be expressed as incompletely-known MDPs. Rather than using static data like the other categories, Reinforcement Learning works with a dynamic dataset from the environment. RL's goal is not to label or cluster data but to map actions from states of the environment to get maximum outcome. Overall, RL is an optimization problem.

The trade-off between exploration and exploitation is one of the biggest challenges in RL. The agent might prefer to choose actions that can produce rewards and explore the effects of other actions rather than choosing one action all the time. Researchers have studied the exploration-exploitation dilemma for a long time. There are methods like epsilon-greedy, optimistic initial values, and Upper-Confidence Bound (UCB) action selection, and each of them has some positive results and some problems [1]. Nevertheless, the dilemma is still a challenge in RL studies.

Another challenge in RL is the Reward Delay. Sometimes, the rewards in the process will not happen instantly; instead, they will return in the long term. Taken the Go game as an example, the only way of expressing the reward itself is winning or losing the game, but making each move will affect the result. That is to say, before jumping into the learning, the reward-delayed system has to be adjusted with a better rewarding solution. Furthermore, this is usually difficult to realize in applications.

To sum up, before making the agent interact with the environment, we need to construct the system first. We first need to confirm whether the environment is actual or simulated. Secondly, a reward function is appropriately set up to incentive the agent to take proper steps. Then, a good policy needs to be chosen with logic and parameters. Next, a training algorithm for the agent is employed. Lastly, the agent should be put into the environment and starts training.

1.3.1. RL Algorithms

RL algorithms might be model-free or model-based, value-based and (or) policy-based, on-policy or off-policy, with sample backups or full backups, as shown in Figures 2 and 3.

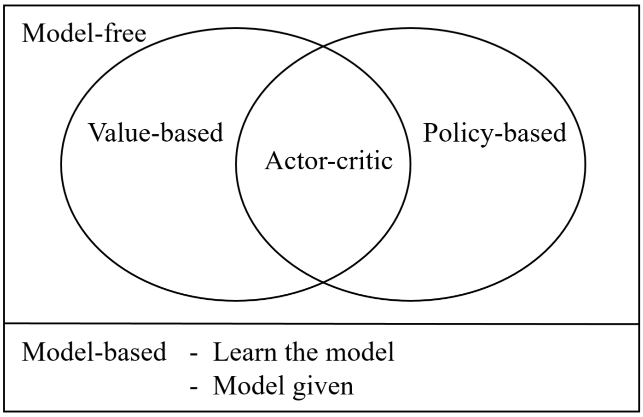

Figure 2. Reinforcement Learning Methods.

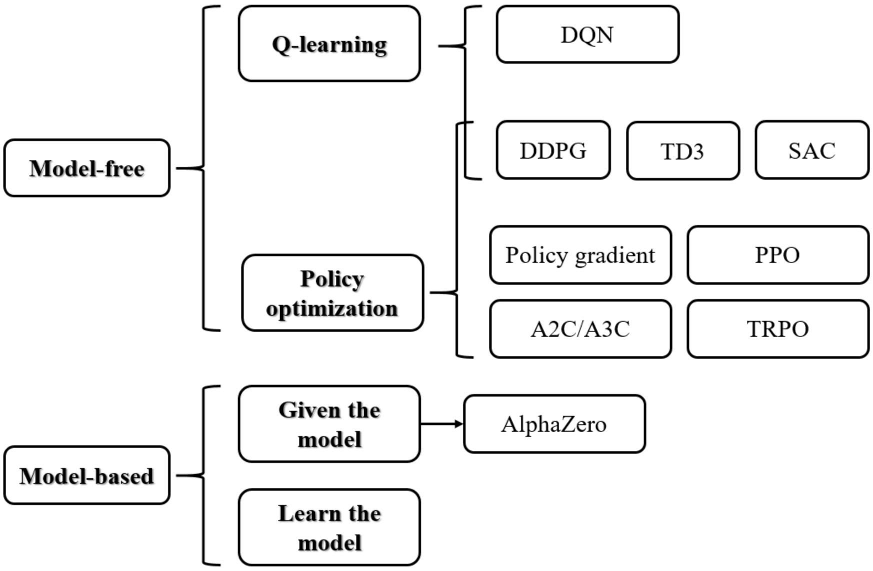

Figure 3. The taxonomy of representative RL Algorithms (For more information, see [2]).

Dynamic Programming

Dynamic Programming (DP) is a general solution for optimization problems with recursion. MDPs with Bellman Equation is a good example. The idea of DP is to store the solutions of subproblems and use them later when needed rather than re-compute them. While solving RL, the solution of the subproblem is the value function of a policy. Thus, the control problem is to find an optimal value function (Value Iteration) or an optimal policy (Policy Iteration).

Recall equations from the MDP section, Policy iteration (PI) integrates policy evaluation and policy improvement. Based on the precious value function, an improved policy can be found. Eventually, it will converge to an optimal policy; Value Iteration (VI) starts with a value function. An improved value function will be obtained in the iterative process until reaching the optimal value function. DP is model-based.

Monte Carlo and TD Learning

Two classic methods to estimate the value of a policy are Monte Carlo (MC) and Temporal Difference (TD) Learning. With the MC-based method, the agent learns the value $V_\pi(s)$ as the average return of all sampled episodes. For each episode, the approximated value can be updated using an incremental mean according to the formula:

$$V(s_t) \leftarrow V(s_t) + \alpha[G_t - V(s_t)] \tag{11}$$

Where $V(s_t)$ is the estimated state value, initialized using a certain strategy; α is a rate that influences the convergence; G_t is the return from time t that the actor first visited the state-action pair (or sum of returns from each time t it visited the pair), can be calculated as below:

$$G_t = r_{t+1} + \gamma r_{t+2} + ... + \gamma^{T-1} r_T \tag{12}$$

MC method updates the value after a complete episode, making it a slow algorithm with significant variance. TD Learning solves this drawback by updating value every time step and does not require a whole episode to update the value. At time $t+1$, the TD approach compares the value with a TD target $r_{t+1} + V(s_{t+1})$ and updates it with TD error. The simplest TD method, TD(0), explains the intuition:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{13}$$

TD methods predict the future before knowing the total accumulated reward—this is a form of bootstrapping. There is no bootstrapping in MC or DP methods. MC and TD methods are both model-free, and they both work with sample backups.

Q-Learning

As learned in the MDP section, Q value is defined as the expected reward when taking action a at state s, following policy π. So Q value can measure which action $a \in \mathit{Actions}$ is better than the others. Thus, we can find a better policy. This method is called Q-learning [3].

The Q-learning method lets the initial policy π interact with the environment. Then the Q value can be obtained at the state s, taking action a. TD or MC methods will now contribute to finding a better policy π' to replace the previous actor. Eventually, an optimal actor will be reached. Q-learning uses a table with the rows as the states (or observations) and the columns as the actions. Each cell in the table is the Q-value of this state, taking this action.

SARSA [4] is similar to Q-learning. However, Q-learning is off-policy, and SARSA is an On-Policy algorithm. SARSA selects a new action, using the same policy, instead of the maximum reward for the next state, to update the Q-values.

Q-learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{\alpha} Q(s',a') - Q(s,a)] \quad (s \leftarrow s') \tag{14}$$

SARSA:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \quad (s \leftarrow s', a \leftarrow a') \tag{15}$$

1.4. Deep Reinforcement Learning

Deep Learning (DL), or a Deep Neural Network (DNN), has more than one hidden layer in a Neural Network. At each hidden layer, nonlinear transformation, or activation function, is applied to obtain a new representation of the input from the previous layer [5]. As a result, deep Learning can interpret the pixels directly.

Deep Reinforcement Learning (DRL) combines Reinforcement Learning and Deep Learning. It is more capable of learning from raw sensors or images as input, enabling end-to-end learning, which opens up more applications in robotics, video games, NLP, computer vision, healthcare, and more.

1.4.1. Value-Based Algorithms

A milestone in value-based DRL is employing Deep Q-Networks (DQN) to play Atari games by Google DeepMind [6] in 2013. In DQN, the agent is trained by Deep Neural Networks rather than the Q-table. In this structure, the current states (or observations) from the environment are inputs, and the outputs are the agent's actions. DQN overcomes unstable learning by four main techniques: Experience Replay, Target Network, Clipping Rewards, and Skipping Frames [7].

In 2015, van Hasselt et al. [8] proposed Double DQN, which uses deep neural networks combined with the idea of Double Q-learning [9], yielding more accurate results on Atari games. Schaul et al. [10] used Prioritized Experience Replay with Double DQN, making learning from experience replay more efficient. In 2016, Wang et al. [11] presented Dueling DQN, which has two streams to estimate state-value and the advantages for each action separately, leading to better policy evaluation in the presence of similar-valued actions. Categorical DQN was introduced in 2017 by Bellemare et al. [12], which models the value distribution within a DQN agent. Noisy DQN from Fortunato et al. [13] has an agent with parametric noise added to its weights and uses stochastic network layers for exploration. Furthermore, Rainbow DQN was introduced in 2018 [14], combining several components in one agent, including DDQN, prioritized replay, dueling networks, multi-step learning [15], Distributional RL, and Noisy Nets.

Other techniques include the General Reinforcement Learning Architecture (Gorila) by Nair et al. [16] that performs asynchronous training for many agents in a distributed setting; Hierarchical DQN introduced by Kulkarni et al. [17] integrates hierarchical value functions and operates at different temporal scales.

1.4.2. Policy-Based Algorithms

A policy gradient algorithm called Trust Region Policy Optimization (TRPO) was introduced by Schulman et al. [18] in 2015. This method updates policies by repeatedly optimizing a local approximation with a KL divergence penalty. It avoids the performance collapse due to the difference of new and old policies in parameter space. Proximal Policy Optimization (PPO) proposed in [19] has the same background as TRPO. They share similar benefits, but it is simpler to implement and has better sample complexity. This variant of PPO does not have a KL-divergence. Instead, it has a specialized clipping in the objective function to avoid the new policy getting far from the old policy.

1.4.3. Actor-Critic Algorithms

Deep DPG (DDPG), based on deterministic policy gradient (DPG), [20], was introduced in [21]. It uses off-policy data and Bellman equation to learn a Q-function and then uses the Q-function to learn a policy. O'Donoghue et al. [22] also combines policy gradient with Q-learning, drawing experience from a replay buffer. This combination allows us to estimate the Q-values from the action preferences of the policy. Then Q-learning updates are applied to the policy.

Soft Actor critic (SAC) [23] is an off-policy actor-critic algorithm based on the maximum entropy RL framework. The actor aims to maximize the expected reward and maximize the entropy simultaneously, such that it achieves fast learning with high randomness in the policy.

Advantage Actor-Critic algorithms have two main variants: the Asynchronous Advantage Actor-Critic (A3C) and the Advantage Actor-Critic (A2C) A3C was introduced in [24], which implements parallel training for multiple agents in parallel environments and updates a global value function.

A table of representative DRL techniques is shown in Table 1.

Table 1. Representative DRL Techniques.

	DRL Algorithms	**Main Techniques**
Value-based	DQN [6]	Experience Replay, Target Network, Clipping Rewards, Clipping Rewards, and Skipping Frames
	Double DQN [8]	Double Q-learning
	Dueling DQN [11]	Dueling Neural Network architecture
	Prioritized DQN [10]	Prioritized experience replay
	Bootstrapped DQN [25]	Deep exploration with DNNs
	Distributional DQN [12]	Distributional Bellman equation
	Noisy DQN [13]	Parametric noise added to weights
	Rainbow DQN [14]	Combine 6 extensions to the DQN
	Hierarchical DQN [17]	Hierarchical value functions
	Gorila [16]	Asynchronous training for multi agents
Policy-based	TRPO [18]	KL divergence constraint
	PPO [19]	Specialized clipping in the objective function
Actor-Critic	Deep DPG [21]	DNN and DPG
	TD3 [26]	Twin Delayed DDPG
	PGQ [22]	Policy gradient and Q-learning
	Soft Actor Critic (SAC) [23]	Maximum entropy RL framework
	A3C [24]	Asynchronous Gradient Descent

2. Applications

Deep Reinforcement Learning has been widely utilized in many domains, as shown in Table 2, which is summarized from [5]. This chapter will discuss the recent advances in applications in games, robotics, Natural Language Processing (NLP), transportation, industrial applications, communication and networking, and more topics. Usually, different techniques may be employed in an application besides DRL, e.g., AlphaGo is trained using supervised learning and reinforcement learning. In this report, POMDP problems are taken into more consideration, but in reality, it is hard to define an application as a firm POMDP problem or not. We focus applications generally based on Deep Reinforcement Learning, not only on POMDP problems.

Table 2. Various DRL Applications, summarized from [5].

Domains	Applications
Healthcare	DTRs, HER/EMR, diagnosis
Education	Educational games, recommendation, proficiency estimation
Transportation	Traffic control
Energy	Decision control
Finance	Trading, risk management
Science, Engineering, Art	Math, Physics, music, animation
Business management	Recommendation, customer management
Computer systems	Resourse management, security
Games	Board games, card games, video games
Robotics	Sim-to-real, control
Computer vision	Recognition, detection, perception
NLP	Sequence generation, translation, dialogue

2.1. Games

Researchers have continuously been studying RL algorithms with games as testbeds since long ago. This section discusses this research in board games, card games, and video games. This section refers to Li's overview [5].

2.1.1. Board Games

In the board games like Go, chess, checkers, and Backgammon, players have perfect information about each other, and the environment is deterministic (not MDPs). The AI research in two-person games can trace back to the 1950s, Claude Shannon and Alan Turing proposed computer programs that could challenge humans.

RL in Board Games

There have been significant steps in AI playing games towards the DRL direction. In 1994, TD-Gammon [27] was a Neural Network to estimate evaluation function with TD learning for Backgammon. Buro [28] also wrote papers proposing methods of playing Othello programs in the 1990s, for further references, see: https://skatgame.net/mburo/publications.html, accessed on 11 January 2021. Schaeffer et al. [29] developed Chinook, playing checkers, between 1989 and 2007. Ginsberg [30] is the inventor of GIB for playing the computer bridge. For the Hex games, Huang employed MCTS in MoHex [31], and Gao et al. [32] trained the neural networks by canonical maximum likelihood to enhance MoHex 2.0.

Mannen's thesis [33] implemented experiments in the training of neural networks as evaluation functions for a chess program. Block et al. [34] used reinforcement learning in chess engines, extending KinghtCap's [35] learning algorithm with a larger database. Later, Lai [36] presented Giraffe, using Deep Reinforcement Learning with automatic feature extraction in chess, outperformed other chess engines during the time.

AlphaGo

In 2016, AlphaGo [37] became the first computer Go program that won the professional Go player Lee Sedol. This is the historical moment that AI proves to be able to surpass human learning in board games. AlphaGo was trained by a combination of supervised learning and reinforcement learning of self-play. The networks also combined Monte Carlo simulation with value and policy networks. Later, Silver et al. [38] introduced ALphaGo Zero, based solely on RL without human data or guidance beyond game rules. A neural network is trained to predict AlphaGo or other winner's action selections. This resulted in higher quality and more robust self-play features. Later, Silver et al. [39] also came up with a general algorithm called AlphaZero. It realized superhuman performance in many other games and defeated champion programs in chess, shogi, and Go. Furthermore, Kimura et al. [40] applied AlphaZero to POMDP maze problems, and it turned out effective.

AlphaGo is a turning point in the application of Deep Reinforcement Learning. It has made tremendous progress in recent years.

2.1.2. Card Games

Card games, like many variants of Poker, UNO, and mahjong, have incomplete information. The agents and the environment are stochastic and uncertain. These games are usually POMDP problems.

Texas Hold'em Poker

One of the classic poker games is Texas Hold'em Poker. Davidson [41], Davidson et al. [42], Felixa and Reis [43], and Bayes' Bluff presented by Southey et al. [44] used Neural Networks to model opponents in Texas Hold'em Poker, thus the network was able to predict the next actions of players correctly.

Zinkevich et al. [45] presented Counterfactual Regret Minimization(CFR). CFR is to compute an approximate Nash equilibrium for the abstract game by iteratively traversing the game tree. Based on the CFR technique, Bowling et al. [46] weakly solved Heads-up Limit Hold'em Poker. Furthermore, Deep CFR was discussed by Brown et al. [47].

DeepStack computer program presented by Moravčík et al. [48], uses deep neural networks as the value function for its depth-limited look-ahead, defeated professional poker players for the first time, in heads-up no-limit Texas Hold'em. More information about DeepStack can be found at https://www.deepstack.ai/, accessed on 11 January 2021.

Prior breakthroughs have been limited to settings with two players. Brown and Sandholm [49] proposed Pluribus, a superhuman AI capable of defeating elite human professionals in six-player no-limit Texas hold'em poker, with self-play through improved Monte Carlo CFR.

Hearts and Wizard

Hearts is a multi-player non-cooperative finite-state zero-sum imperfect-information game that can formulate a POMDP [50]. Fujita and Ishii [51] developed model-based RL for large-scale multi-player games with partial observability and applied it to Hearts, using Markov Chain Monte Carlo (MCMC) technique. Sturtevant and White [52] developed a player for the game of hearts of 4 players, based on stochastic linear regression and TD learning.

In 2017, Wagenaar's thesis [53] used a multi-layer perceptron (MLP) and Monte Carlo learning to play Game Hearts and realized DRL. However, more work is required to realize mature performance in the Hearts game.

Similar to Hearts, Wizard is also a multi-player imperfect-information poker game. RL has turned out effective for these POMDP games. Backhus et al. [54] applied RL in the Wizard game, separately dealing with forecasting and trick playing.

RLCard

In recent years, the full-width extensive-form fictitious play (XFP) was introduced by Heinrich and Heinrich and Lanctot [55], Neural Fictitious Self-Play (NFSP) was presented by Heinrich and Heinrich and Silver [56], and Unified Game-Theoretic Approach was proposed by Lanctot et al. [57]. We can see that RL strategies can perform well in betting games. These results give the inspiration for researchers to explore DRL in more card games.

Zha et al. [58] presented RLCard, an open-source toolkit for reinforcement learning research in card games. It provides various card game environments, including Blackjack, Leduc Hold'em, Texas Hold'em, UNO, Dou Dizhu, and Mahjong, with easy-to-use interfaces (see Figure 4).

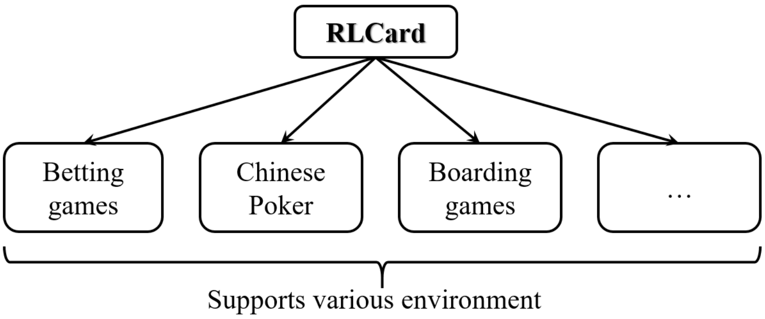

Figure 4. RLCard: It supports various styles of card games and board games [58].

2.1.3. Video Games

Video games are valuable resources with video frames as inputs to RL/AI agents, in which computer games are the most popular testbeds for DRL algorithms. Similar to the MDP process as shown in Figure 1, DRL for video games employs CNNs to extract features from the video frames (environment) to be recognized by the agents. Recently, DRL has been applied in Sega, Nintendo, Xbox, etc. Shao et al. [59] made a comprehensive survey of DRL in video games. Based on which, Figure 5 presents popular video-game platforms for RL research. In this section, we particularly dig into details about games with POMDP characteristics.

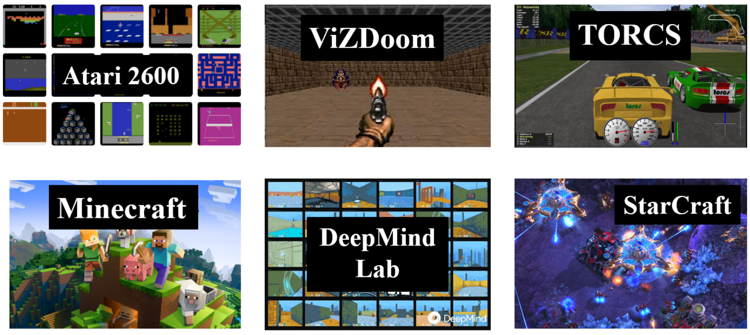

Figure 5. Popular video-game platforms for Reinforcement Learning research, summarized from [59].

Atari

A traditional platform, the Arcade Learning Environment (ALE) [60], provides an interface to lots of games in Atari 2600 and is a pioneer evaluation platform in RL research. As described in Chapter One, many value-based methods are tested in Atari 2600 and the un-mentioned techniques, such as DRQN [61] and ADRQN [62]. Aytar et al. [63] presented an exciting method of playing Atari exploration games by watching Youtube. They map unaligned videos from several sources to a common representation, then embed a single YouTube video in this representation to construct a reward function. Hence, the agent learns to imitate human gameplay. This method exceeds human-level performance on MONTEZUMA'S REVENGE, PITFALL! and PRIVATE EYE, with no environment rewards in games.

ViZDoom

Doom is a classical first-person shooter (FPS) game with a 3D environment, where agents observe perspective 2D projections from their positions as pixel matrices. The partial observability of game states is the main challenge. ViZDoom [64] is a Doom-based platform that offers API.

Lample and Chaplot [65] used a divide and conquered method in Doom to divide the action space. They trained an action agent and a navigation agent. Different agents will act [66] based on the detection of an enemy. Akimov and Makarov [67] proposed

Categorical 51 with Multi-step (C51M) architecture, outperformed baseline methods in ViZDoom environment.

In the survey [59], they mentioned the following important techniques that evaluated in ViZDoom: Wu and Tian [68] proposed a framework combining A3C with curriculum learning [69]; Parisotto and Salakhutdinov [70] developed Neural Map, a memory system with an adaptable write operator, using a spatially structured 2D memory image to store arbitrary environment information; Shao et al. [71] also introduced ACKTR, which teaches agents to battle in the ViZDoom environment.

TORCS

The Open Racing Car Simulator (TORCS) is a racing game with acceleration, braking, and steering as actions. It has been used as testbeds for visual-input games. It is also an essential platform for testing autonomous driving methods with its realistic feature.

In 2014, Koutník et al. [72,73] had Max-Pooling Convolutional Neural Network (MPCNN) compressor and recurrent neural network (RNN) controllers evolved successfully to control a car in TORCS based on visual input; DDPG [21], as mentioned, was assessed in TORCS; Wang et al. [74] adopted DDPG and selected sensor information from TORCS as inputs, evaluated on different modes in TORCS; Sallab et al. [75] formulated two categories of algorithms for lane-keeping assist: DQN for discrete actions and Deep Deterministic Actor-Critic Algorithm (DDAC) for continuous actions; Furthermore, Liu et al. [76] proposed DDPG method with expert demonstrations and supervised loss, simulated in TORCS, which improves efficiency and stability significantly; Verma et al. [77] presented Programmatically Interpretable Reinforcement Learning (PIRL), to generate interpretable agent policies and a new method called Neurally Directed Program Search (NDPS), to find a programmatic policy with maximal reward;

In addition to the works shown above, some techniques discussed by Shao et al. [59]: Sharma et al. [78] proposed Fine Grained Action Repetition (FiGAR) to improve DDPG; Gao et al. [79] used TORCS to evaluate Normalized Actor-Critic (NAC); Mazumder et al. [80] developed a property called state-action permissibility (SAP), with which the agent learns to solve the lane-keeping problem in TORCS, with faster training process; Li et al. [81] breaks the vision-based lateral control system down into a perception module and a control module; Zhu and Zhao [82] uses DRL to train a CNN to perceive data.

Minecraft

Minecraft is a well-known sandbox video game where players can explore a blocky, procedurally-generated 3D world and build structures and earthworks with freedom in different game modes. It supports scenarios from navigation and survival to collaboration and problem-solving tasks, making it a rich environment to carry out RL.

Project Malmo [83] is an AI experimentation platform on top of Minecraft, created by Microsoft. Marlo, based on Malmo, provides a higher-level API and a more standardized RL-friendly environment for RL study. In 2019, Guss et al. [84] presented a comprehensive, large-scale, simulator-paired dataset of human demonstrations: MineRL, which provides an RL environment and competitions.

Some RL tasks created in Minecraft were effectively achieved. Liu et al. [85] constructed tasks with shared characteristics in Minecraft for multi-task DQN (MDQN), which turned out effective. Oh et al. [86] also designed tasks to evaluate DRL architectures: Memory Q-Network (MQN), Recurrent Memory Q-Network (RMQN), and Feedback Recurrent Memory Q-Network (FRMQN). Frazier [87] conducted experiments with two RL algorithms: Feedback Arbitration and Newtonian Action Advice, enabling human teachers to give action advice. Furthermore, HogRider [88] is the champion agent of MCAC in 2017, using a Q-learning approach with state-action abstraction and warm start; Tessler et al. [89] proposed a lifelong learning system that is able to transfer knowledge from one task to another, incorporated with Hierarchical-DRLN with a deep skill array and skill distillation;

Jin et al. [90] utilized counterfactual regret minimization to update an approximation to an advantage-like function iteratively.

DeepMind Lab

DeepMind Lab [91] is a 3D customizable first-person game platform for agent-based AI research, a modified version of Quake III Arena. It is usually used to study how RL agents deal with complex tasks in large, partially observed, and visually diverse worlds.

Many researchers have created RL tasks in DeepMind Lab and tested algorithms. Johnston [92] introduced an unsupervised auxiliary task—“surprise pursuit”. It quantifies the surprise” the agent encounters when exploring the environment and trains a policy to maximize this “surprise”.

There also are researchers who used DM-Lab to propose new architectures. As mentioned by [59], Mankowitz et al. [93] proposed Unicorn, which demonstrates robust continual learning; Schmitt et al. [94] had a teacher agent to kickstart the training of a new student agent. The method mentioned by Jaderberg et al. [95] trains plenty of agents concurrently from lots of parallel matches, where agents play cooperatively in teams, competing in randomly generated environments.

Later on, DeepMind also invented Psychlab [96], a psychophysics testing room embedded inside the world of DM-Lab. This framework allows DRL agents to be directly compared with humans, lifted from cognitive psychology and visual psychophysics.

Real-Time Strategy Games

Real-time strategy games (RTS) allow players to play the game in “real-time” simultaneously rather than to take turns to play. RTS like StarCraft, Dota, and LOL are popular platforms in AI research. The real-time games usually consist of thousands of time steps and hundreds of actions selected in real-time through the gameplay. StarCraft is a game in which players balance high-level economic decisions in the control of tons of units. In each time step, the agent receives imperfect information from observations. In 2019, DeepMind proposed AlphaStar [97], which defeats professional players for the first time. Table 3 presents the techniques carried out in StarCraft.

MOBA (Multiplayer Online Battle Arena) games have two teams, and each team consists of five players. OpenAI Five [98] used Dota2 as the research platform and achieved expert-level performance, and is the first AI system who could defeat the world champions in an esports game.

Table 3. Techniques used in StarCraft, summarized based on [59].

StarCraft Algorithms	Main Techniques
GMEZO [99]	Micromanagement; greedy MDP; episodic zero-order exploration
BiCNet [100]	Multiagent bidirectionally coordinated network
Stabilising Experience Replay [101]	Importance sampling; fingerprints
PS-MAGDS [102]	Parameter sharing multi-agent gradient descent SARSA(λ)
MS-MARL [103]	Master-slave architecture; multi-agent
QMIX [104]	Decentralised policies in a centralised end-to-end training
NNFQ & CNNFQ [105]	Neural network fitted Q-Learning
SC2LE [106]	FullyConv-A2C; baseline results
Relational DRL [107]	Self-attention to iteratively reason about the relations
TStarBots [108]	Flat action structure; hierarchical action structure
Modular architecture [109]	Splits responsibilities between multiple modules
Two-level hierarchical RL [110]	Macro-actions; a two-layer hierarchical architecture

2.2. Robotics

Robotics is widely used in science, industries, healthcare, education, entertainment, and many more domains. Robotics has many challenging problems involved, including perceptron, control system, operation system, etc. The development of Deep Reinforcement Learning paves the way in robotics. Many applications mentioned in the gaming section, including AlphaGo, are also involved in Robotics. While in this section, we focus on manipulation and locomotion.

For more surveys in Robotics, see [111] in 2009, a study of robot learning from demonstration (LfD), with which a policy is learned from demonstrations, provided by a teacher; Deisenroth [112] made a survey on policy search for robotics in 2013; In 2014, Kober and Peters [113] provided a general survey on RL in robotics; Tai et al. [114] presented a comprehensive survey for learning control in robotics from reinforce to imitation in 2018.

A 3D physics simulator MuJoCo physics engine [115], which stands for Multi-Joint dynamics with Contact, was developed by OpenAI for continuous control. Mainly based on which series algorithms are evaluated.

2.2.1. Manipulation

Dexterous manipulation is one of the most challenging control problems in robotics due to the barriers of high dimensionality, intermittent contact dynamics, and underactuation in dynamic object manipulation; Dexterous manipulation can be widely used in different domains. These make it a hot topic in Robotics. Nguyen and La [116] provided a review of DRL for Robot Manipulation with the challenges, the existing problems, and future research directions.

For reinforcement learning, researchers made significant progress, making DRL in manipulation tasks accessible in recent years. For example, in 2008, Peters and Schaal [117] evaluated policy gradient methods on parameterized motor primitives, and Episodic Natural Actor-Critic [118] outperforms the others; In 2010, Theodorou et al. [119] presented Policy Improvement with Path Integrals (PI2); In 2010, Peters et al. [120] suggested Relative Entropy Policy Search (REPS) method; In 2011, Kalakrishnan et al. [121] used PI2 with force control policies. Nair et al. [122] demonstrated overcoming the exploration problem and solved the block stacking task using RL.

Multi-Fingered Hands

Dexterous multi-fingered hands can perform a wide range of manipulation skills. They are essential in research while it requires high dimensional observation and ample action space. Kumar et al. [123] demonstrated online planning based on an ADROIT platform, and Kumar et al. [124] described a model-based method combining optimizing Linear–Gaussian controllers models with KL-constrained optimization. It was applied on a pneumatically-actuated tendon-driven 24-DoF hand, which learned hand manipulation involving clockwise rotation of the object.

Rajeswaran et al. [125] then demonstrated that model-free DRL could scale up to complex manipulation tasks. It does not require a model of the dynamics and can optimize the policy directly. They used a combination of RL and imitation learning Demo Augmented Policy Gradient (DAPG), achieved in object relocation, in-hand manipulation, door opening, and tool use tasks.

However, in the real world, the interaction time with the environment is limited, the model-free methods perform poorly in such cases. Haarnoja et al. [126] studied how maximum entropy policies trained using soft Q-learning. Policies learned using soft Q-learning can be composed to create new policies, the optimality of which can be enclosed knowing the divergence between the composing policies. By composing existing skills can efficiently improve the training process. This method is sample efficient than prior model-free DRL methods and can be performed for simulated and real-world tasks.

Zhu et al. [127] surveyed multi-fingered manipulation and used model-free DRL algorithms with low-cost hardware, succeeded in learning complex contact-rich behaviors.

Considering interacting with fragile objects, Huang et al. [128] realized gentle object manipulation, minimizing impact forces, with dynamics-based surprise and penalty-based surprise.

Grasping

Levine et al. [129] trained using guided policy search (GPS), which can perform policy search by supervised learning, where supervision is granted by a simple trajectory-centric RL method. It was evaluated in various tasks, like learning to insert, screw, fit, and place the objects. DRL has also been applied to learn grasping objects [130,131], Popov et al. [132] introduced two extensions to the Deep Deterministic Policy Gradient (DDPG) algorithm to improve data efficiency with linear speedup for 16 parallel workers. The method succeeded in grasping objects and precisely stacking them on another in simulation. Kalashnikov et al. [133] introduced QT-Opt, a scalable self-supervised RL framework, enables closed-loop vision-based control. They achieved real-world grasping tasks with a high success rate based only on RGB vision perceptron.

Opening a Door

Gu et al. [134] presented Asynchronous NAF with practical extensions. The experiment shows success in several 3D manipulation skills in simulation and a door opening skill on real robots.

Cloth Manipulation

Regarding cloth manipulation, the state of the cloth can vary, unlike rigid objects. Tsurumine et al. [135] proposed Deep P-Network (DPN) and Dueling Deep P-Network (DDPN) to combine smooth policy update with feature extraction, evaluated in flipping a handkerchief and folding a t-shirt by a dual-arm robot. Jangir et al. [136] investigated the effectiveness of different textile state representations and the importance of speed and trajectory, solved dynamic cloth manipulation tasks by restricting the manipulator workspace.

Magnetic Manipulation

Magnets can be utilized in magnetic resonance imaging (MRI) applications, lifting machinery, levitation devices, jewelry, bearings, audio equipment, and hard disc drives. de Bruin et al. [137] deployed two experience replay databases in the method, one filled with experiences in the First In First Out (FIFO) manner. The experiences are overwritten with new experiences in the other. This method limits the detrimental effects of exploration, tested on a simulated horizontal magnetic manipulation task.

2.2.2. Locomotion

Over the recent years, many RL methods have been proposed to control locomotion, locomotion examples are shown in Figure 6. On humanoid platforms such as Nao, Pepper, Asimo, and Robotis-op2 include the cameras [138], many algorithms were evaluated. The main goal for robot locomotion is to imitate human motions dynamically. Several main techniques such as DeepDPG [21], A3C [24], TRPO [18] etc., that were previously covered, were tested in robotics locomotion and turned out effective. They are the key to the vast development of DRL algorithms in locomotion [139,140].

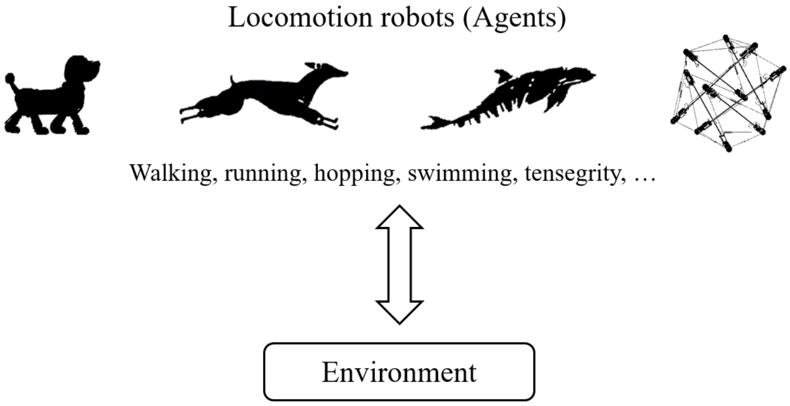
Locomotion robots (Agents)
Walking, running, hopping, swimming, tensegrity, …
Environment

Figure 6. Locomotion examples.

In Robotics, many tasks are complex and poorly specified. Christiano et al. [141] had humans compare possible trajectories, used the data to learn a reward function, and optimized the learned reward function with RL, and scaled up this method to DRL. In particular, the robotics tasks including walker, hopper, swimmer, etc., were implemented in MujoCo and OpenAI Gym [142].

DeepLoco [143] adopted a two-level hierarchical control framework. First, low-level controllers (LLC) learn to operate at a timescale and achieve robust walking gaits. Second, high-level controllers (HLC) learn the policy at the timescale of steps by invoking desired step targets for the low-level controller. Both levels of the hierarchy use an identical style of the actor-critic algorithm.

To develop sound strategies for integrating or merging policies for multiple skills, Berseth et al. [144] proposed Progressive Learning and Integration via Distillation (PLAiD). Distillation refers to the combination of the policies of one or more experts, thus creating one controller to perform the tasks of a set of experts. The main blocks are policy transfer and multi-task policy distillation, evaluated on five different terrains with a 2D PD-biped. Compared with MultiTasker, PLAiD scales better.

Generally, model-free methods can learn a wide range of robotic skills but require a large number of samples to achieve good performance. Model-based algorithms can provide more efficient learning but are challenging to scale up to high-capacity models. Md-Mf [145] uses deep neural network dynamics models to initialize a model-free learner, then uses the policy trained by a model-based controller as the initial policy for a TRPO (model-free) algorithm. Testing locomotion skills on a simulated quadrupedal robot, this method surpasses the individual model-free and model-based approach.

Duan et al. [146] studied Truncated Natural Policy Gradient (TNPG) and compared it with TRPO and DDPG. They also provided a benchmark of continuous control problems for RL.

Following are some examples of different locomotion skills.

Walking/Running

Xie et al. [147] demonstrated DRL on a bipedal robot Cassie, developed by Agility Robotics. The framework used PPO and PD control on a simulated model of Cassie in the MuJoCo [115] simulation and is robust to disturbances.

Designing agile locomotion for robots usually requires human expertise and manual tuning. Tan et al. [148] presented a system to learn quadruped locomotion from scratch or allow users to guide the learning process. In addition, it narrows the gap between simulation and the real world, successfully developed two gaits, trotting and galloping.

Haarnoja et al. [140] proposed an extension of the soft actor-critic (SAC) algorithm with dynamic temperature, based on maximum entropy RL. It was applied to a real-world

Minitaur robot without any model or simulation. This sample-efficient and stable algorithm outperforms standard SAC, DDPG, TD3, and PPO.

Özalp et al. [138] evaluated Double Dueling Q Networks (D3QN) [149] and DQN [7] for locomotion using a Robotis-op2 humanoid robot.

Hopping

Peng et al. [150] introduced a mixture of actor-critic experts (MACE), constructed of multiple actor-critic pairs where each specializes in particular aspects of the motion. It can work with high-dimensional state descriptions for terrain-adaptive dynamic locomotion skills. With leaps or steps as output actions, the method was tested on seven classes of terrain and gives excellent results.

Swimming

Heess et al. [151] surveyed a hierarchical motor control architecture that encourages the low-level controller to focus on reactive motor control. The high-level controller directs behavior towards the task goal by transmitting a modulatory signal. Evaluated on a swimming snake, a quadruped, and a humanoid, this method failed in end-to-end learning but turned out effective in transferring tasks with sparse reward.

To learn stochastic energy-based policies for continuous states and actions, not only in tabular domains, Haarnoja et al. [152] introduced Soft Q-learning, which expresses the optimal policy via a Boltzmann distribution. The simulated swimming snake and quadrupedal robot results show that the method can effectively capture complex multi-modal behaviors and learn general-purpose stochastic policies.

To narrow the gap between simulation and the real world and generalize from training to testing, Pinto et al. [153] introduced robust adversarial reinforcement learning (RARL). The adversary learns to apply destabilizing forces on specific points on the system, encouraging the protagonist to retain a strong control policy. This method improves training stability, and is robust to training/testing conditions, and outperforms the baselines even without an adversary.

Verma et al. [154] studied that DRL can learn efficient collective swimming in unsteady and vortical flow fields.

Tensegrity

Tensegrity robots are composed of rigid rods connected by elastic cables, which display high tolerance to physical damage. With their unique properties, tensegrity robots are appealing for planetary exploration rovers. The primary locomotion of tensegrity robots is rolling.

Zhang et al. [155] applied an extension of mirror descent guided policy search (MDGPS) to periodic locomotion movements on a SUPERball tensegrity robot. The learned locomotion policies performed well, are more effective, and can generalize better than open-loop policies. Luo et al. [156] showed that MDGPS could efficiently learn even with limited sensory inputs.

2.2.3. Robotics Simulators

Robotics simulators are used to simulate and model the structures or actions of a physical robot without depending on the real-world machine. The use of simulators to develop robots allows for programs to be written and debugged offline, and the final versions can then be transferred to the real robot, thus saving cost and time. Table 4 presents the popular robotics simulators nowadays.

Table 4. Robotics simulators, adapted from Wikipedia (2021) [157].

Software	Developers	Main Language	Physics Engine	Robot Family
Gazabo	Open Source Robotics Foundation (OSRF)	C++	ODE/Bullet/Simbody/DART	UGV, UAV, AUV, humanoid, etc.
RoboDK	RoboDK [158]	Python	Gravity plugin	Robot arms
SimSpark		C++, Ruby	ODE	UGV, humanoid, etc.
Webots	Cyberbotics Ltd.	C++	Fork of ODE	UGV, UAV, AUV, humanoid, etc.
OpenRAVE	OpenRAVE Community	C++, Python	ODE/Bullet	UGV, humanoid, etc.

2.3. Natural Language Processing

Natural language processing (NLP) focuses on the communication between computers and humans—how to program computers to understand, analyze, and process natural language, including generation, language grounding, information extraction, dialogue, etc. Due to the unique features of NLP, DRL has been playing an important role. For more surveys, see [5,159–163].

2.3.1. Neural Machine Translation

Neural Machine Translation (NMT) refers to using machine learning software to translate content into another language. RL has been shown as an effective technique for NMT systems recently. Examples of machine translation providers are Google Translate, Yandex Translate, Watson Language Translator, etc. Wu et al. [164] provided a literature review on RL for NMT, in which they compared several vital factors, including baseline reward and reward shaping, and studied how to train better NMT systems. They also proposed training with source-side and (or) target-side monolingual data and succeeded in the Chinese-English translation task.

He et al. [165] proposed dual-NMT for English-French dual translation. The English-to-French translation (primal) agent and French-to-English translation (dual) agent can form a closed loop and generate feedback signals to train the models by teaching each other through the RL process. This system works well on bilingual translation, especially by learning from monolingual data.

In the traditional NMT setting, the translator has to wait until the user completes the sentence, then translates. Satija and Pineau [166] presented an approach to interpret in real-time.

The open-source toolkit OpenNMT was described by Klein et al. [167], that prioritizes efficiency and modularity; Zhang et al. [168] introduced THUMT that supports minimum risk training and semi-supervised training.

For generative adversarial net (GAN) in NMT, BR-CSGAN [169] leverages the BLEU reinforced GAN to improve the NMT, and Wu et al. [170] introduced Adversarial-NMT with a CNN based adversary.

2.3.2. Dialogue

Dialogue systems generally have two categories—task-oriented dialog agents and chatbots. Task-oriented dialog agents usually have short Q-and-A conversations to help complete specific tasks or provide information, while chatbots are usually designed human-likely and used for entertainment [171].

Cuayáhuitl has done significant research on dialogue systems with RL [172–175]. In 2017, he presented SimpleDS [176], as shown in Figure 7. It's a publicly available dialogue system trained with DQN, which extends the traditional methods [175] and avoids manual feature engineering by directly selecting actions from raw text and user responses.

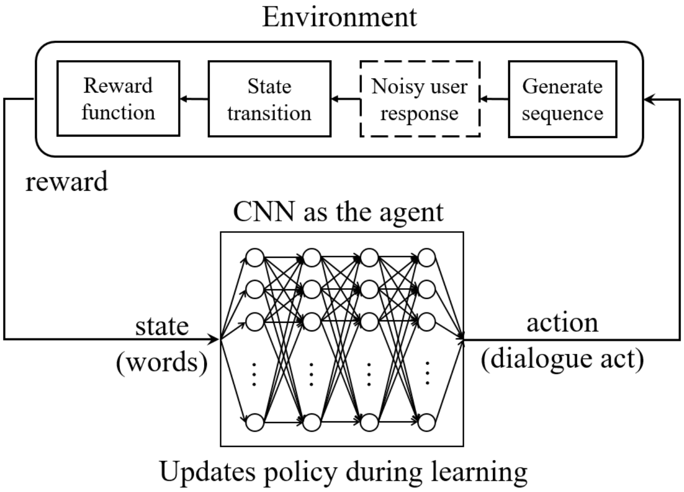

Figure 7. SimpleDS architecture [176].

Cuayáhuitl et al. [177] proposed a two-stage method—the first stage reduces the times of weight updates, and the second uses limited mini-batches sampled from experience replay for accelerating the induction of single or multi-domain dialogue policies. The training turns out five times faster than a baseline. Furthermore, Cuayáhuitl et al. [178] described an ensemble-based approach applied to value-based DRL chatbots. Actions are derived from sentence clustering, the training datasets are derived from dialogue clustering, and the reward function is derived from human–human dialogues without manual annotations. The ensemble of DRL agents turns out more promising than a single DRL agent or SEQ2SEQ model.

Most DRL systems perform poorly in exploring online in the environment or learn effectively from off-policy data. Jaques et al. [179] leveraged models pre-trained on data with KL-control to penalize divergence. They used uncertainty estimates with dropout to lower bound the target Q-values. Tested in the real world with humans, the Way Off-policy algorithm can effectively learn from different reward functions from human interaction data and outperforms prior methods in off-policy batch RL.

Text Generation

Text generation is the basis of dialogue systems. Ranzato et al. [180] proposed mixed incremental cross-entropy reinforce (MIXER) with a loss function combining REINFORCE [181] and cross-entropy. For sequence prediction, Bahdanau et al. [182] proposed an actor-critic algorithm, inspired by Ranzato et al. [180]. Yu et al. [183] introduced sequence generative adversarial nets with policy gradient (SeqGAN). Li et al. [184] from Huawei Technologies proposed a new DL framework for paraphrase generation, consisting of a generator and an evaluator. The generator is modeled as the SEQ2SEQ learning model for paraphrase generation. As a deep matching model, the evaluator is trained via supervised learning or inverse RL in different settings for paraphrase identification.

Sentence Simplification

Sentence simplification is also crucial in dialogue systems, which aims to make sentences easier to read and understand for the agents. Zhang and Lapata [185] addressed this simplification issue with an encoder-decoder model with a DRL framework, known as Deep Reinforcement Sentence Simplification (DRESS). Their work succeeded in simplicity, grammaticality, and semantic fidelity to the input.

Coherent Dialogues

Some neural models of dialogue generation tend to be short-sighted, and modeling the future direction of dialogue is crucial to generating coherent and exciting conversations. Li et al. [186] applied DRL with policy search and curriculum learning on the SEQ2SEQ model [187], which can generate utterances that optimize future reward.

Goal-Oriented Dialogues

Dhingra et al. [188] proposed KB-InfoBot, an end-to-end dialogue agent to help people search Knowledge Bases (KBs). They offered a differentiable probabilistic framework to compute the posterior distribution (a Soft-KB lookup) to avoid symbolic queries used in previous works.

To evaluate persuasion strategies and negotiation dialogue, Keizer et al. [189] presented a comparative evaluation of various negotiation strategies on the game "Settlers of Catan" online. It outperforms the original rule-based strategy and human players.

Ilievski et al. [190] presented goal-oriented chatbot dialog management bootstrapping with transfer learning, which helps users achieve a predefined goal. Due to the domain specificity, the available data is limited to obtain. The transfer learning method is to boost the performances and mitigate the limitation. They describe that transfer learning improves performance significantly with a low number of goals or whole target domain data.

For social robot navigation, Ciou et al. [191] presented composite reinforcement learning (CRL) framework, with which agents learn social navigation with sensor input and reward update based on human feedback.

2.3.3. Visual Dialogue

Das et al. [192] introduced a goal-driven training for visual question answering (VQA) and dialogue agents. The experiment contains two cooperative robots, with one giving descriptions and the other giving answers. The results show that it outperforms supervised learning agents.

Thanks to the encoder-decoder architectures, End-to-end dialogue systems have developed rapidly. However, there are some drawbacks in encoder-decoder models when engaging in task-oriented dialogue. Strub et al. [193] introduced the DRL method to optimize vision-based task-oriented dialogues with REINFORCE algorithm. It resulted in significant improvement over a supervised baseline model, which was capable of generating coherent dialogue.

Visual Captioning

Evaluation metrics are created to evaluate image captioning. Standard syntactic evaluation metrics, such as BLEU, METEOR, and ROUGE, are usually not well correlated. SPICE and CIDEr are better correlated but are hard to optimize. Liu et al. [194] used a policy gradient method to optimize a linear combination of SPICE and CIDEr (SPIDEr).

In reference to visual problems, image captioning is challenging due to the complexity of understanding the image content and describing it in natural language. Xu et al. [195] introduced attention-based model for image caption generation. Adaptive encoder-decoder attention model [196] with an LSTM extension as a visual sentinel decides whether to attend to the image and where to generate information. Ren et al. [197] introduced a decision-making framework for image captioning, which has a policy network and a value network. Both networks are learned using an actor-critic approach with a visual-semantic embedding reward.

Wang et al. [198] proposed an Adversarial Reward Learning (AREL) framework for visual storytelling. The model learns a reward function from human demonstrations, then optimizes policy search with the learned reward function.

Video captioning is to generate a description of the actions in the video automatically. Wang et al. [199] proposed a hierarchical RL framework for video captioning. A high-level module learns to create sub-goals, and a low-level module recognizes the actions to fulfill the sub-goal. The HRL model turns out a good performance on MSR-VTT and Charades Captions datasets. For video captioning, Pasunuru and Bansal [200] presented a mixed-loss policy gradient approach that allows for metric-based optimization, then improved with an entailment-corrected CIDEnt reward, successfully achieved on MSR-VTT.

Visual Relationship and Attribute Detection

Apart from generating labels and captioning, understanding relationships among objects and their attributes in the scene is challenging for computers. Liang et al. [201] proposed a deep Variation-structured Reinforcement Learning (VRL) framework for visual relationship and attribute detection. A directed action graph is built to represent semantic correlations between object categories, predicates, and attributes. Then based on the state information, a variation-structured traversal over the chart is used to generate an action set for each step. Predictions are made using DRL and the extracted phrases in the state vector. This method can significantly achieve better detection results on datasets involving relationships and attributes.

3. Conclusions

3.1. Summary

In this paper, we have presented the fundamental concepts of Markov Decision Processes(MDP), Partially Observable Markov Decision Processes (POMDPs), Reinforcement Learning (RL), and Deep Reinforcement Learning (DRL). Generally, we have discussed the representative DRL techniques and algorithms. The core elements of an RL algorithm are value function, policy, reward, model and planning, exploration, and knowledge. In DRL applications that were discussed, most of the algorithms applied are extensions of the representative algorithms, for example, DQN, DPG, A3C, etc., with some optimizations, respectively. We then discussed the applications of DRL in some popular domains. As DRL provides optimal policies and strategies, it has been widely utilized in gaming, including almost all kinds of games. As the leading entertainment among people, a better gaming experience is always desired. Another popular application of DRL is in Robotics. We discussed manipulation, locomotion, and other aspects of robotics where DRL can be applied. Robotics simulators are also discussed, where DRL is used to select the appropriate simulator when needed. Finally, the application of DRL in natural language processing (NLP) is presented.

3.2. Final Thoughts

As we witnessed, DRL has been developing fast and on enormous scales. Some groups pushed forward the development of DRL, like OpenAI, DeepMind, AI research office in Alberta, and research center led by Rich Sutton, and more. From the research, we have acknowledged the algorithms; variations and the development of all the optimizations. Based on the review, we are more familiar with the usability of each method. For example, model-free methods work better in scaled-up tasks. The value function is key to RL, in DQN and its many extensions; policy gradient approaches have also been utilized in lots of applications, such as robotics, dialogue systems, machine translation, etc. New learning techniques have also been applied, such as transfer, curriculum learning, adversarial networks, etc. When we select a method or combinations of methods to apply in our models, the characteristics we need to consider are stability, convergence, generalizing power, accuracy, efficiency, scalability, robustness, safety, cost, speed, simplicity, etc. For example, in healthcare, we focus more on stability and security rather than simplicity or cost. This topic, along with applications in transportation, communications and networking, and industries, will be discussed in a Part 2 follow-up paper.

Author Contributions: Conceptualization, X.X. and S.F.; methodology, X.X.; formal analysis, X.X.; investigation, X.X. and S.F.; resources, X.X. and S.F.; writing—original draft preparation, X.X.; writing—review and editing, X.X. and S.F.; visualization, X.X.; supervision, X.X. and S.F.; funding acquisition, S.F. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The authors would like to express their appreciation to friends and colleagues who had assisted in the preparation of this paper.

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

References

1. Sutton, R.S.; Barto, A.G. *Reinforcement Learning—An Introduciton*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018.
2. OpenAI. Part 2: Kinds of RL Algorithms—A Taxonomy of RL Algorithms. 2018. Available online: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html (accessed on 1 July 2021).
3. Watkins, C.J.; Dayan, P. Q Learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
4. Rummery, G.A.; Niranjan, M. *Q-Learning Using Connectionist Systems*; University of Cambridge, Department of Engineering: Cambridge, UK, 1994
5. Li, Y. Deep Reinforcement Learning: An Overview. *arXiv* **2018**, arXiv:1701.07274.
6. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
7. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
8. van Hasselt, H.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double Q-Learning. *arXiv* **2015**, arXiv:1509.06461
9. van Hasselt, H. Double Q-Learning. *Adv. Neural Inf. Process. Syst.* **2010**, *23*, 2613–2621.
10. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. *arXiv* **2016**, arXiv:1511.05952.
11. Wang, Z.; Schaul, T.; Hessel, M.; can Hasselt, H.; Lanctot, M.; de Freitas, N. Dueling Network Architectures for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016
12. Bellemare, M.G.; Dabney, W.; Munos, R. A Distributional Perspective on Reinforcement Learning. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
13. Fortunato, M.; Azar, M.G.; Piot, B. Noisy Networks for Exploration. *arXiv* **2018**, arXiv:1706.10295.
14. Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining Improvements in Deep Reinforcement Learning. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018
15. Sutton, R.S. Learning to Predict by the Methods of Temporal Differences. *Mach. Learn.* **1988**, *3*, 9–44. [CrossRef]
16. Nair, A.; Srinivasan, P.; Blackwell, S.; Alcicek, C.; Fearon, R.; Maria, A.D.; Panneershelvam, V.; Suleyman, M.; Beattie, C.; Petersen, S.; et al. Massively Parallel Methods for Deep Reinforcement Learning. *arXiv* **2015**, arXiv:1507.04296.
17. Kulkarni, T.D.; Narasimhan, K.R.; Saeedi, A.; Tenenbaum, J.B. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. *arXiv* **2016**, arXiv:1604.06057.
18. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.; Abbeel, P. Trust Region Policy Optimization. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015
19. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
20. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic Policy Gradient Algorithms. In Proceedings of the 31st International Conference on Machine Learning, Bejing, China, 22–24 June 2014
21. Lillicrap, T.P.; Hunt, J.J.; Alexander Pritzel, N.H.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1509.02971.
22. O'Donoghue, B.; Munos, R.; Kavukcuoglu, K.; Mnih, V. Combining policy gradient and Q-learning. *arXiv* **2017**, arXiv:1611.01626.
23. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018
24. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Harley, T.; Lillicrap, T.P.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016
25. Osband, I.; Blundell, C.; Pritzel, A.; Van Roy, B. Deep Exploration via Bootstrapped DQN. *arXiv* **2016**, arXiv:1602.04621.
26. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv* **2018**, arXiv:1802.09477.
27. Tesauro, G. TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play. *Neural Comput.* **1994**, *6*, 215–219. [CrossRef]
28. Buro, M. How Machines have Learned to Play Othello. *IEEE Intell. Syst.* **1999**, *14*, 12–14
29. Schaeffer, J.; Burch, N.; Bjornsson, Y.; Kishimoto, A.; Muller, M.; Lake, R.; Lu, P.; Sutphen, S. Checkers Is Solved. *Science* **2007**, *317*, 1518–1522. [CrossRef]

30. Ginsberg, M.L. GIB: Imperfect Information in a Computationally Challenging Game. *J. Artif. Intell. Res.* **2001**, *14*, 303–358. [CrossRef]
31. Huang, S.C.; Arneson, B.; Hayward, R.B.; Müller, M.; Pawlewicz, J. MoHex 2.0: A Pattern-Based MCTS Hex Player. In *Computers and Games*; van den Herik, H.J., Iida, H., Plaat, A., Eds.; Series Title: Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2014; Volume 8427, pp. 60–71. [CrossRef]
32. Gao, C.; Hayward, R.; Muller, M. Move Prediction Using Deep Convolutional Neural Networks in Hex. *IEEE Trans. Games* **2018**, *10*, 336–343. [CrossRef]
33. Mannen, H.; Wiering, M. Learning To Play Chess Using Reinforcement Learning with Database Games. Master's Thesis, Cognitive Artificial Intelligence Utrecht University, Utrecht, The Netherlands, October 2003
34. Block, M.; Bader, M.; Tapia, E.; Ramírez, M.; Gunnarsson, K.; Cuevas, E.; Zaldivar, D.; Rojas, R. Using Reinforcement Learning in Chess Engines. *Res. Comput. Sci.* **2008**, *35*, 31–40.
35. Baxter, J.; Tridgell, A.; Weaver, L. KnightCap: A chess program that learns by combining TD(lambda) with game-tree search. *arXiv* **1999**, arXiv:cs/9901002
36. Lai, M. Giraffe: Using Deep Reinforcement Learning to Play Chess. *arXiv* **2008**, arXiv:1509.01549.
37. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
38. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of Go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef]
39. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **2018**, *362*, 1140–1144. [CrossRef] [PubMed]
40. Kimura, T.; Sakamoto, K.; Sogabe, T. Development of AlphaZero-based Reinforcment Learning Algorithm for Solving Partially Observable Markov Decision Process (POMDP) Problem. *Bull. Netw. Comput. Syst. Softw.* **2020**, *9*, 69–73.
41. Davidson, A. Using Artifical Neural Networks to Model Opponents in Texas Hold'em. Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, November 1999. Available online: http://www.spaz.ca/aaron/poker/nnpoker.pdf (accessed on 1 July 2021)
42. Davidson, A.; Billings, D.; Schaeffer, J.; Szafron, D. Improved Opponent Modeling in Poker. Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, January 2002. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.480.8603&rep=rep1&type=pdf (accessed on 1 July 2021)
43. Felixa, D.; Reis, L.P. An Experimental Approach to Online Opponent Modeling in Texas Hold'em Poker. In *Advances in Artificial Intelligence—SBIA 2008*; Zaverucha, G., da Costa, A.L., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 83–92.
44. Southey, F.; Bowling, M.; Larson, B.; Piccione, C.; Burch, N.; Billings, D.; Rayner, D.C. Bayes' Bluff: Opponent Modelling in Poker. *arXiv* **2012**, arXiv:1207.1411
45. Zinkevich, M.; Johanson, M.; Bowling, M.; Piccione, C. Regret Minimization in Games with Incomplete Information. In *Advances in Neural Information Processing Systems 20*; Platt, J.C., Koller, D., Singer, Y., Roweis, S.T., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2008; pp. 1729–1736.
46. Bowling, M.; Burch, N.; Johanson, M.; Tammelin, O. Heads-up limit hold'em poker is solved. *Science* **2015**, *347*, 145–149. [CrossRef] [PubMed]
47. Brown, N.; Lerer, A.; Gross, S.; Sandholm, T. Deep Counterfactual Regret Minimization. *arXiv* **2018**, arXiv:1811.00164.
48. Moravčík, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; Bowling, M. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* **2017**, *356*, 508–513. [CrossRef] [PubMed]
49. Brown, N.; Sandholm, T. Superhuman AI for multiplayer poker. *Science* **2019**, *365*, 885–890. [CrossRef]
50. Ishii, S.; Fujita, H.; Mitsutake, M.; Yamazaki, T.; Matsuda, J.; Matsuno, Y. A Reinforcement Learning Scheme for a Partially-Observable Multi-Agent Game. *Mach. Learn.* **2005**, *59*, 31–54. [CrossRef]
51. Fujita, H.; Ishii, S. Model-based Reinforcement Learning for Partially Observable Games with Sampling-based State Estimation. *Neural Comput.* **2007**, *19*, 3051–3087. doi:10.1162/neco.2007.19.11.3051. [CrossRef]
52. Sturtevant, N.R.; White, A.M. Feature Construction for Reinforcement Learning in Hearts. In *International Conference on Computers and Games*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 122–134.
53. Wagenaar, M.; Wiering, D.M. Learning to Play the Game of Hearts Using Reinforcement Learning and a Multi-Layer Perceptron. Bachelor's Thesis, University of Groningen, Groningen, The Netherlands, 2017. Available online: http://fse.studenttheses.ub.rug.nl/id/eprint/15440 (accessed on 11 July 2021).
54. Backhus, J.C.; Nonaka, H.; Yoshikawa, T.; Sugimoto, M. Application of reinforcement learning to the card game Wizard. In Proceedings of the 2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE), Tokyo, Japan, 1–4 October 2013; pp. 329–333.
55. Heinrich, J.; Lanctot, M. Fictitious Self-Play in Extensive-Form Games. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 805–813
56. Heinrich, J.; Silver, D. Deep Reinforcement Learning from Self-Play in Imperfect-Information Games. *arXiv* **2016**, arXiv:1603.01121

57. Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Perolat, J.; Silver, D.; Graepel, T. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. In *Advances in Neural Information Processing Systems 30*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; pp. 4190–4203.
58. Zha, D.; Lai, K.H.; Cao, Y.; Huang, S.; Wei, R.; Guo, J.; Hu, X. RLCard: A Toolkit for Reinforcement Learning in Card Games. *arXiv* **2019**, arXiv:1910.04376
59. Shao, K.; Tang, Z.; Zhu, Y.; Li, N.; Zhao, D. A Survey of Deep Reinforcement Learning in Video Games. *arXiv* **2019**, arXiv:1912.10944
60. Bellemare, M.G.; Naddaf, Y.; Veness, J.; Bowling, M. The Arcade Learning Environment: An Evaluation Platform for General Agents. *J. Artif. Intell. Res.* **2013**, *47*, 253–279. [CrossRef]
61. Hausknecht, M.; Stone, P. Deep Recurrent Q-Learning for Partially Observable MDPs. *arXiv* **2017**, arXiv:1507.06527.
62. Zhu, P.; Li, X.; Poupart, P.; Miao, G. On Improving Deep Reinforcement Learning for POMDPs. *arXiv* **2018**, arXiv:1704.07978.
63. Aytar, Y.; Pfaff, T.; Budden, D.; Paine, T.; Wang, Z.; de Freitas, N. Playing hard exploration games by watching YouTube. In *Advances in Neural Information Processing Systems 31*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; pp. 2930–2941.
64. Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; Jaskowski, W. ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. In Proceedings of the 2016 IEEE Conference on Computational Intelligence and Games (CIG), Santorini, Greece, 20–23 September 2016.
65. Lample, G.; Chaplot, D.S. Playing FPS Games with Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1609.05521.
66. Papoudakis, G.; Chatzidimitriou, K.C.; Mitkas, P.A. Deep Reinforcement Learning for Doom using Unsupervised Auxiliary Tasks. *arXiv* **2018**, arXiv:1807.01960.
67. Akimov, D.; Makarov, I. Deep Reinforcement Learning with VizDoom First-Person Shooter. *CEUR Workshop Proc.* **2019**, *2479*, 3–17.
68. Wu, Y.; Tian, Y. Training Agent for First-Person Shooter Game with Actor-Critic Curriculum Learning. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
69. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum Learning. In Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, Montreal, QC, Canada, 14–18 June 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 41–48. [CrossRef]
70. Parisotto, E.; Salakhutdinov, R. Neural Map: Structured Memory for Deep Reinforcement Learning. *arXiv* **2017**, arXiv:1702.08360.
71. Shao, K.; Zhao, D.; Li, N.; Zhu, Y. Learning Battles in ViZDoom via Deep Reinforcement Learning. In Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games (CIG), Maastricht, The Netherlands, 14–17 August 2018; pp. 1–4
72. Koutník, J.; Schmidhuber, J.; Gomez, F. Online Evolution of Deep Convolutional Network for Vision-Based Reinforcement Learning. In Proceedings of the International Conference on Simulation of Adaptive Behavior, Aberystwyth, UK, 23–26 August 2014; pp. 260–269. [CrossRef]
73. Koutník, J.; Schmidhuber, J.; Gomez, F. Evolving deep unsupervised convolutional networks for vision-based reinforcement learning. In Proceedings of the 2014 Conference on Genetic and Evolutionary Computation—GECCO '14, Vancouver, BC, Canada, 12–16 July 2014; ACM Press: Vancouver, BC, Canada, 2014; pp. 541–548. [CrossRef]
74. Wang, S.; Jia, D.; Weng, X. Deep Reinforcement Learning for Autonomous Driving. *arXiv* **2019**, arXiv:1811.11329.
75. Sallab, A.E.; Abdou, M.; Perot, E.; Yogamani, S. End-to-End Deep Reinforcement Learning for Lane Keeping Assist. *arXiv* **2016**, arXiv:1612.04340.
76. Liu, K.; Wan, Q.; Li, Y. A Deep Reinforcement Learning Algorithm with Expert Demonstrations and Supervised Loss and its application in Autonomous Driving. In Proceedings of the 2018 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; pp. 2944–2949.
77. Verma, A.; Murali, V.; Singh, R.; Kohli, P.; Chaudhuri, S. Programmatically Interpretable Reinforcement Learning. *arXiv* **2019**, arXiv:1804.02477.
78. Sharma, S.; Lakshminarayanan, A.S.; Ravindran, B. Fine grained action repetition for deep reinforcement learning. *arXiv* **2017**, arXiv:1702.06054.
79. Gao, Y.; Xu, H.; Lin, J.; Yu, F.; Levine, S.; Darrell, T. Reinforcement Learning from Imperfect Demonstrations. *arXiv* **2018**, arXiv:1802.05313.
80. Mazumder, S.; Liu, B.; Wang, S.; Zhu, Y.; Liu, L.; Li, J. Action Permissibility in Deep Reinforcement Learning and Application to Autonomous Driving. In *KDD'18 Deep Learning Day*; ACM: London, UK, 2018.
81. Li, D.; Zhao, D.; Zhang, Q.; Chen, Y. Reinforcement Learning and Deep Learning based Lateral Control for Autonomous Driving. *arXiv* **2018**, arXiv:1810.12778
82. Zhu, Y.; Zhao, D. Driving Control with Deep and Reinforcement Learning in The Open Racing Car Simulator. In *Neural Information Processing*; Cheng, L., Leung, A.C.S., Ozawa, S., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 326–334.
83. Johnson, M.; Hofmann, K.; Hutton, T.; Bignell, D. The Malmo Platform for Artificial Intelligence Experimentation. IJCAI. 2016. pp. 4246–4247. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1045.1281&rep=rep1&type=pdf (accessed on 1 July 2021).

84. Guss, W.H.; Houghton, B.; Topin, N.; Wang, P.; Codel, C.; Veloso, M.; Salakhutdinov, R. MineRL: A Large-Scale Dataset of Minecraft Demonstrations. *arXiv* **2019**, arXiv:1907.13440.
85. Liu, L.T.; Dogan, U.; Hofmann, K. Decoding multitask DQN in the world of Minecraft. In Proceedings of the 13th European Workshop on Reinforcement Learning (EWRL), Barcelona, Spain, 3–4 December 2016; p. 11
86. Oh, J.; Chockalingam, V.; Singh, S.P.; Lee, H. Control of Memory, Active Perception, and Action in Minecraft. *arXiv* **2016**, arXiv:1605.09128
87. Frazier, S. Improving Deep Reinforcement Learning in Minecraft with Action Advice. *arXiv* **2019**, arXiv:1908.01007.
88. Xiong, Y.; Chen, H.; Zhao, M.; An, B. HogRider: Champion Agent of Microsoft Malmo Collaborative AI Challenge. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
89. Tessler, C.; Givony, S.; Zahavy, T.; Mankowitz, D.J.; Mannor, S. A Deep Hierarchical Approach to Lifelong Learning in Minecraft. *arXiv* **2016**, arXiv:1604.07255
90. Jin, P.H.; Levine, S.; Keutzer, K. Regret Minimization for Partially Observable Deep Reinforcement Learning. *arXiv* **2017**, arXiv:1710.11424
91. Beattie, C.; Leibo, J.Z.; Teplyashin, D.; Ward, T.; Wainwright, M.; Küttler, H.; Lefrancq, A.; Green, S.; Valdés, V.; Sadik, A.; et al. DeepMind Lab. *arXiv* **2016**, arXiv:1612.03801.
92. Johnston, L. Surprise Pursuit Auxiliary Task for Deepmind Lab Maze. n.d. p. 7. Available online: http://cs231n.stanford.edu/reports/2017/pdfs/610.pdf (accessed on 1 July 2021).
93. Mankowitz, D.J.; Zídek, A.; Barreto, A.; Horgan, D.; Hessel, M.; Quan, J.; Oh, J.; van Hasselt, H.; Silver, D.; Schaul, T. Unicorn: Continual Learning with a Universal, Off-policy Agent. *arXiv* **2018**, arXiv:1802.08294
94. Schmitt, S.; Hudson, J.J.; Zídek, A.; Osindero, S.; Doersch, C.; Czarnecki, W.M.; Leibo, J.Z.; Küttler, H.; Zisserman, A.; Simonyan, K.; et al. Kickstarting Deep Reinforcement Learning. *arXiv* **2018**, arXiv:1803.03835.
95. Jaderberg, M.; Czarnecki, W.M.; Dunning, I.; Marris, L.; Lever, G.; Castañeda, A.G.; Beattie, C.; Rabinowitz, N.C.; Morcos, A.S.; Ruderman, A.; et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv* **2018**, arXiv:1807.01281.
96. Leibo, J.Z.; de Masson d'Autume, C.; Zoran, D.; Amos, D.; Beattie, C.; Anderson, K.; Castañeda, A.G.; Sanchez, M.; Green, S.; Gruslys, A.; et al. Psychlab: A Psychology Laboratory for Deep Reinforcement Learning Agents. *arXiv* **2018**, arXiv:1801.08116.
97. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [CrossRef]
98. Open, A.I.; Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; et al. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1912.06680.
99. Usunier, N.; Synnaeve, G.; Lin, Z.; Chintala, S. Episodic Exploration for Deep Deterministic Policies: An Application to StarCraft Micromanagement Tasks. *arXiv* **2016**, arXiv:1609.02993.
100. Peng, P.; Yuan, Q.; Wen, Y.; Yang, Y.; Tang, Z.; Long, H.; Wang, J. Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games. *arXiv* **2017**, arXiv:1703.10069.
101. Foerster, J.N.; Nardelli, N.; Farquhar, G.; Torr, P.H.S.; Kohli, P.; Whiteson, S. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. *arXiv* **2017**, arXiv:1702.08887.
102. Shao, K.; Zhu, Y.; Zhao, D. StarCraft Micromanagement With Reinforcement Learning and Curriculum Transfer Learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **2019**, *3*, 73–84. [CrossRef]
103. Kong, X.; Xin, B.; Liu, F.; Wang, Y. Revisiting the Master-Slave Architecture in Multi-Agent Deep Reinforcement Learning. *arXiv* **2017**, arXiv:1712.07305.
104. Rashid, T.; Samvelyan, M.; de Witt, C.S.; Farquhar, G.; Foerster, J.N.; Whiteson, S. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv* **2018**, arXiv:1803.11485.
105. Tang, Z.; Zhao, D.; Zhu, Y.; Guo, P. Reinforcement Learning for Build-Order Production in StarCraft II. In Proceedings of the 2018 Eighth International Conference on Information Science and Technology (ICIST), Cordoba/Granada/Seville, Spain, 30 June–6 July 2018; pp. 153–158.
106. Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A.S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J.P.; Schrittwieser, J.; et al. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv* **2017**, arXiv:1708.04782.
107. Zambaldi, V.F.; Raposo, D.; Santoro, A.; Bapst, V.; Li, Y.; Babuschkin, I.; Tuyls, K.; Reichert, D.P.; Lillicrap, T.P.; Lockhart, E.; et al. Relational Deep Reinforcement Learning. *arXiv* **2018**, arXiv:1806.01830.
108. Sun, P.; Sun, X.; Han, L.; Xiong, J.; Wang, Q.; Li, B.; Zheng, Y.; Liu, J.; Liu, Y.; Liu, H.; et al. TStarBots: Defeating the Cheating Level Builtin AI in StarCraft II in the Full Game. *arXiv* **2018**, arXiv:1809.07193.
109. Lee, D.; Tang, H.; Zhang, J.O.; Xu, H.; Darrell, T.; Abbeel, P. Modular Architecture for StarCraft II with Deep Reinforcement Learning. *arXiv* **2018**, arXiv:1811.03555.
110. Pang, Z.J.; Liu, R.Z.; Meng, Z.Y.; Zhang, Y.; Yu, Y.; Lu, T. On reinforcement learning for full-length game of starcraft. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 4691–4698.
111. Argall, B.D.; Chernova, S.; Veloso, M.; Browning, B. A survey of robot learning from demonstration. *Robot. Auton. Syst.* **2009**, *57*, 469–483. [CrossRef]
112. Deisenroth, M.P. A Survey on Policy Search for Robotics. *Found. Trends Robot.* **2013**, *2*, 1–142. [CrossRef]

113. Kober, J.; Peters, J. Reinforcement Learning in Robotics: A Survey. In *Learning Motor Skills: From Algorithms to Robot Experiments*; Springer International Publishing: Cham, Switzerland, 2014; pp. 9–67. [CrossRef]
114. Tai, L.; Zhang, J.; Liu, M.; Boedecker, J.; Burgard, W. A Survey of Deep Network Solutions for Learning Control in Robotics: From Reinforcement to Imitation. *arXiv* **2015**, arXiv:1612.07139.
115. Todorov, E.; Erez, T.; Tassa, Y. Mujoco: A physics engine for model-based control. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems,Vilamoura/Algarve, Portugal, 7–12 October 2012; pp. 5026–5033.
116. Nguyen, H.; La, H. Review of Deep Reinforcement Learning for Robot Manipulation. In Proceedings of the 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 25–27 February 2019; pp. 590–595.
117. Peters, J.; Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural Netw.* **2008**, *21*, 682–697. [CrossRef]
118. Peters, J.; Schaal, S. Natural actor-critic. *Neurocomputing* **2008**, *71*, 1180–1190. [CrossRef]
119. Theodorou, E.; Buchli, J.; Schaal, S. Reinforcement learning of motor skills in high dimensions: A path integral approach. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010; pp. 2397–2403.
120. Peters, J.; Mülling, K.; Altun, Y. Relative entropy policy search. In Proceedings of the AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010; Volume 10, pp. 1607–1612.
121. Kalakrishnan, M.; Righetti, L.; Pastor, P.; Schaal, S. Learning force control policies for compliant manipulation. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 4639–4644.
122. Nair, A.; McGrew, B.; Andrychowicz, M.; Zaremba, W.; Abbeel, P. Overcoming Exploration in Reinforcement Learning with Demonstrations. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6292–6299.
123. Kumar, V.; Tassa, Y.; Erez, T.; Todorov, E. Real-time behaviour synthesis for dynamic hand-manipulation. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 6808–6815.
124. Kumar, V.; Todorov, E.; Levine, S. Optimal control with learned local models: Application to dexterous manipulation. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 378–383.
125. Rajeswaran, A.; Kumar, V.; Gupta, A.; Schulman, J.; Todorov, E.; Levine, S. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. *arXiv* **2017**, arXiv:1709.10087.
126. Haarnoja, T.; Pong, V.; Zhou, A.; Dalal, M.; Abbeel, P.; Levine, S. Composable Deep Reinforcement Learning for Robotic Manipulation. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6244–6251.
127. Zhu, H.; Gupta, A.; Rajeswaran, A.; Levine, S.; Kumar, V. Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 3651–3657.
128. Huang, S.H.; Zambelli, M.; Kay, J.; Martins, M.F.; Tassa, Y.; Pilarski, P.M.; Hadsell, R. Learning Gentle Object Manipulation with Curiosity-Driven Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1903.08542.
129. Levine, S.; Finn, C.; Darrell, T.; Abbeel, P. End-to-End Training of Deep Visuomotor Policies. *arXiv* **2015**, arXiv:1504.00702.
130. Pinto, L.; Gupta, A. Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours. *arXiv* **2015**, arXiv:1509.06825.
131. Levine, S.; Pastor, P.; Krizhevsky, A.; Quillen, D. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *arXiv* **2016**, arXiv:1603.02199.
132. Popov, I.; Heess, N.; Lillicrap, T.P.; Hafner, R.; Barth-Maron, G.; Vecerík, M.; Lampe, T.; Tassa, Y.; Erez, T.; Riedmiller, M.A. Data-efficient Deep Reinforcement Learning for Dexterous Manipulation. *arXiv* **2017**, arXiv:1704.03073.
133. Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. *arXiv* **2018**, arXiv:1806.10293.
134. Gu, S.; Holly, E.; Lillicrap, T.; Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Marina Bay Sands, Singapore, 29 May–3 June 2017; pp. 3389–3396.
135. Tsurumine, Y.; Cui, Y.; Uchibe, E.; Matsubara, T. Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation. *Robot. Auton. Syst.* **2019**, *112*, 72–83. [CrossRef]
136. Jangir, R.; Alenya, G.; Torras, C. Dynamic Cloth Manipulation with Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1910.14475.
137. de Bruin, T.; Kober, J.; Tuyls, K.; Babuška, R. Improved deep reinforcement learning for robotics through distribution-based experience retention. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 3947–3952.
138. Özalp, R.; Kaymak, C.; Yildirum, Ö.; Ucar, A.; Demir, Y.; Güzeliş, C. An Implementation of Vision Based Deep Reinforcement Learning for Humanoid Robot Locomotion. In Proceedings of the 2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), Sofia, Bulgaria, 3–5 July 2019; pp. 1–5.
139. Danel, M. Reinforcement learning for humanoid robot control. *POSTER May* **2017**.

140. Haarnoja, T.; Zhou, A.; Ha, S.; Tan, J.; Tucker, G.; Levine, S. Learning to Walk via Deep Reinforcement Learning. *arXiv* **2018**, arXiv:1812.11103.
141. Christiano, P.F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; Amodei, D. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems 30*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; pp. 4299–4307.
142. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
143. Peng, X.B.; Berseth, G.; Yin, K.; Van De Panne, M. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Trans. Graph.* **2017**, *36*. [CrossRef]
144. Berseth, G.; Xie, C.; Cernek, P.; Van de Panne, M. Progressive reinforcement learning with distillation for multi-skilled motion control. *arXiv* **2018**, arXiv:1802.04765.
145. Nagabandi, A.; Kahn, G.; Fearing, R.S.; Levine, S. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 7559–7566.
146. Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; pp. 1329–1338.
147. Xie, Z.; Berseth, G.; Clary, P.; Hurst, J.; van de Panne, M. Feedback Control For Cassie With Deep Reinforcement Learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1241–1246.
148. Tan, J.; Zhang, T.; Coumans, E.; Iscen, A.; Bai, Y.; Hafner, D.; Bohez, S.; Vanhoucke, V. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. *arXiv* **2018**, arXiv:1804.10332
149. Huang, Y.; Wei, G.; Wang, Y. V-D D3QN: The Variant of Double Deep Q-Learning Network with Dueling Architecture. In Proceedings of the 2018 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; pp. 9130–9135.
150. Peng, X.B.; Berseth, G.; Van de Panne, M. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Trans. Graph. (TOG)* **2016**, *35*, 1–12 [CrossRef]
151. Heess, N.; Wayne, G.; Tassa, Y.; Lillicrap, T.P.; Riedmiller, M.A.; Silver, D. Learning and Transfer of Modulated Locomotor Controllers. *arXiv* **2016**, arXiv:1610.05182.
152. Haarnoja, T.; Tang, H.; Abbeel, P.; Levine, S. Reinforcement Learning with Deep Energy-Based Policies. *arXiv* **2017**, arXiv:1702.08165.
153. Pinto, L.; Davidson, J.; Sukthankar, R.; Gupta, A. Robust Adversarial Reinforcement Learning. *arXiv* **2017**, arXiv:1703.02702.
154. Verma, S.; Novati, G.; Koumoutsakos, P. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proc. Natl. Acad. Sci. USA* **2018**, *115*, 5849–5854. [CrossRef]
155. Zhang, M.; Geng, X.; Bruce, J.; Caluwaerts, K.; Vespignani, M.; SunSpiral, V.; Abbeel, P.; Levine, S. Deep reinforcement learning for tensegrity robot locomotion. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 634–641.
156. Luo, J.; Edmunds, R.; Rice, F.; Agogino, A.M. Tensegrity Robot Locomotion Under Limited Sensory Inputs via Deep Reinforcement Learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6260–6267.
157. Wikipedia. Robotics Simulator. Available online: https://en.wikipedia.org/wiki/Robotics_simulator (accessed on 1 July 2021).
158. Obst, O.; Rollmann, M. Spark—A Generic Simulator for Physical Multi-Agent Simulations. In *Multiagent System Technologies. MATES 2004*; Lecture Notes in Computer Science; Springer: Berlin, Heidelberg, 2004; Volume 3187. [CrossRef]
159. Wang, W.Y.; Li, J.; He, X. Deep reinforcement learning for NLP. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, Melbourne, Australia, 15–20 July 2018; pp. 19–21.
160. Sharma, A.R.; Kaushik, P. Literature survey of statistical, deep and reinforcement learning in natural language processing. In Proceedings of the 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 5–6 May 2017; pp. 350–354.
161. Xiong, C. Recent Progress in Deep Reinforcement Learning for Computer Vision and NLP. In Proceedings of the 2017 Workshop on Recognizing Families in the Wild, RFIW ’17, New York, NY, USA, 27 October 2017; Association for Computing Machinery; p. 1. [CrossRef]
162. Lapan, M. *Deep Reinforcement Learning Hands-On: Apply Modern RL Methods, with Deep Q-Networks, Value Iteration, Policy Gradients, TRPO, AlphaGo Zero and More*; Packt Publishing Ltd.: Birmingham, UK, 2018.
163. Gao, J.; Galley, M.; Li, L. Neural approaches to conversational AI. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 1371–1374.
164. Wu, L.; Tian, F.; Qin, T.; Lai, J.; Liu, T. A Study of Reinforcement Learning for Neural Machine Translation. *arXiv* **2018**, arXiv:1808.08866.
165. He, D.; Xia, Y.; Qin, T.; Wang, L.; Yu, N.; Liu, T.Y.; Ma, W.Y. Dual Learning for Machine Translation. In *Advances in Neural Information Processing Systems 29*; Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2016; pp. 820–828.

166. Satija, H.; Pineau, J. Simultaneous machine translation using deep reinforcement learning. In Proceedings of the ICML 2016 Workshop on Abstraction in Reinforcement Learning, International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.
167. Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; Rush, A.M. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *arXiv* **2017**, arXiv:1701.02810.
168. Zhang, J.; Ding, Y.; Shen, S.; Cheng, Y.; Sun, M.; Luan, H.; Liu, Y. THUMT: An Open Source Toolkit for Neural Machine Translation. *arXiv* **2017**, arXiv:1706.06415.
169. Yang, Z.; Chen, W.; Wang, F.; Xu, B. Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets. *arXiv* **2017**, arXiv:1703.04887.
170. Wu, L.; Xia, Y.; Tian, F.; Zhao, L.; Qin, T.; Lai, J.; Liu, T.Y. Adversarial neural machine translation. In Proceedings of the 10th Asian Conference on Machine Learning, PMLR, Beijing, China, 14–16 November 2018; pp. 534–549.
171. Jurafsky, D.; Martin, J.H. *Speech & Language Processing—Third Edition Draft*; Pearson Education India: New Delhi, India, 2019.
172. Cuayáhuitl, H.; Renals, S.; Lemon, O.; Shimodaira, H. Evaluation of a hierarchical reinforcement learning spoken dialogue system. *Comput. Speech Lang.* **2010**, *24*, 395–429. [CrossRef]
173. Cuayáhuitl, H.; Dethlefs, N. Spatially-aware dialogue control using hierarchical reinforcement learning. *Acm Trans. Speech Lang. Process. (TSLP)* **2011**, *7*, 1–26. [CrossRef]
174. Cuayáhuitl, H.; Kruijff-Korbayová, I.; Dethlefs, N. Nonstrict hierarchical reinforcement learning for interactive systems and robots. *Acm Trans. Interact. Intell. Syst. (TiiS)* **2014**, *4*, 1–30. [CrossRef]
175. Cuayáhuitl, H.; Keizer, S.; Lemon, O. Strategic Dialogue Management via Deep Reinforcement Learning. *arXiv* **2015**, arXiv:1511.08099.
176. Cuayáhuitl, H. SimpleDS: A Simple Deep Reinforcement Learning Dialogue System. In *Dialogues with Social Robots: Enablements, Analyses, and Evaluation*; Jokinen, K., Wilcock, G., Eds.; Springer Singapore: Singapore, 2017; pp. 109–118. [CrossRef]
177. Cuayáhuitl, H.; Yu, S. Deep reinforcement learning of dialogue policies with less weight updates. In Poceedings of the Interspeech 2017, Stockholm, Sweden, 20–24 August 2017.
178. Cuayáhuitl, H.; Lee, D.; Ryu, S.; Cho, Y.; Choi, S.; Indurthi, S.; Yu, S.; Choi, H.; Hwang, I.; Kim, J. Ensemble-based deep reinforcement learning for chatbots. *Neurocomputing* **2019**, *366*, 118–130. [CrossRef]
179. Jaques, N.; Ghandeharioun, A.; Shen, J.H.; Ferguson, C.; Lapedriza, À.; Jones, N.; Gu, S.; Picard, R.W. Way Off-Policy Batch Deep Reinforcement Learning of Implicit Human Preferences in Dialog. *arXiv* **2019**, arXiv:1907.00456.
180. Ranzato, M.; Chopra, S.; Auli, M.; Zaremba, W. Sequence level training with recurrent neural networks. *arXiv* **2015**, arXiv:1511.06732.
181. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
182. Bahdanau, D.; Brakel, P.; Xu, K.; Goyal, A.; Lowe, R.; Pineau, J.; Courville, A.; Bengio, Y. An actor-critic algorithm for sequence prediction. *arXiv* **2016**, arXiv:1607.07086.
183. Yu, L.; Zhang, W.; Wang, J.; Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
184. Li, Z.; Jiang, X.; Shang, L.; Li, H. Paraphrase Generation with Deep Reinforcement Learning. *arXiv* **2017**, arXiv:1711.00279.
185. Zhang, X.; Lapata, M. Sentence simplification with deep reinforcement learning. *arXiv* **2017**, arXiv:1703.10931.
186. Li, J.; Monroe, W.; Ritter, A.; Jurafsky, D.; Galley, M.; Gao, J. Deep Reinforcement Learning for Dialogue Generation. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–4 November 2016. [CrossRef]
187. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *arXiv* **2014**, arXiv:1409.3215.
188. Dhingra, B.; Li, L.; Li, X.; Gao, J.; Chen, Y.; Ahmed, F.; Deng, L. End-to-End Reinforcement Learning of Dialogue Agents for Information Access. *arXiv* **2016**, arXiv:1609.00777.
189. Keizer, S.; Guhe, M.; Cuayáhuitl, H.; Efstathiou, I.; Engelbrecht, K.P.; Dobre, M.; Lascarides, A.; Lemon, O. Evaluating Persuasion Strategies and Deep Reinforcement Learning Methods for Negotiation Dialogue Agents. ACL. In Proceedings of the 15th Conference of the European chapter of the Association for Computational Linguistics, Valencia, Spain, 3–7 April 2017.
190. Ilievski, V.; Musat, C.; Hossmann, A.; Baeriswyl, M. Goal-Oriented Chatbot Dialog Management Bootstrapping with Transfer Learning. *arXiv* **2018**, arXiv:1802.00500.
191. Ciou, P.H.; Hsiao, Y.T.; Wu, Z.Z.; Tseng, S.H.; Fu, L.C. Composite reinforcement learning for social robot navigation. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 2553–2558.
192. Das, A.; Kottur, S.; Moura, J.M.; Lee, S.; Batra, D. Learning cooperative visual dialog agents with deep reinforcement learning. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2951–2960.
193. Strub, F.; De Vries, H.; Mary, J.; Piot, B.; Courville, A.; Pietquin, O. End-to-end optimization of goal-driven and visually grounded dialogue systems. *arXiv* **2017**, arXiv:1703.05423.
194. Liu, S.; Zhu, Z.; Ye, N.; Guadarrama, S.; Murphy, K. Improved Image Captioning via Policy Gradient Optimization of SPIDEr. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.

195. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 2048–2057.
196. Lu, J.; Xiong, C.; Parikh, D.; Socher, R. Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning. *arXiv* **2016**, arXiv:1612.01887.
197. Ren, Z.; Wang, X.; Zhang, N.; Lv, X.; Li, L.J. Deep Reinforcement Learning-Based Image Captioning With Embedding Reward. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
198. Wang, X.; Chen, W.; Wang, Y.; Wang, W.Y. No Metrics Are Perfect: Adversarial Reward Learning for Visual Storytelling. *arXiv* **2018**, arXiv:1804.09160.
199. Wang, X.; Chen, W.; Wu, J.; Wang, Y.F.; Wang, W.Y. Video Captioning via Hierarchical Reinforcement Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
200. Pasunuru, R.; Bansal, M. Reinforced video captioning with entailment rewards. *arXiv* **2017**, arXiv:1708.02300.
201. Liang, X.; Lee, L.; Xing, E.P. Deep variation-structured reinforcement learning for visual relationship and attribute detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 848–857.