Journal of Artificial Intelligence Research 69 (2020) 1421-1471 Submitted 09/2020; published 12/2020

Deep Reinforcement Learning: A State-of-the-Art Walkthrough

Aristotelis Lazaridis ARISLAZA@CSD.AUTH.GR
Aristotle University of Thessaloniki, School of Informatics
Thessaloniki, 54124, Greece

Anestis Fachantidis ANESTIS@MEDOID.AI
Medoid AI
130 Egnatia St, Thessaloniki, 54622, Greece

Ioannis Vlahavas VLAHAVAS@CSD.AUTH.GR
Aristotle University of Thessaloniki, School of Informatics
Thessaloniki, 54124, Greece

Abstract

Deep Reinforcement Learning is a topic that has gained a lot of attention recently, due to the unprecedented achievements and remarkable performance of such algorithms in various benchmark tests and environmental setups. The power of such methods comes from the combination of an already established and strong field of Deep Learning, with the unique nature of Reinforcement Learning methods. It is, however, deemed necessary to provide a compact, accurate and comparable view of these methods and their results for the means of gaining valuable technical and practical insights. In this work we gather the essential methods related to Deep Reinforcement Learning, extracting common property structures for three complementary core categories: a) Model-Free, b) Model-Based and c) Modular algorithms. For each category, we present, analyze and compare state-of-the-art Deep Reinforcement Learning algorithms that achieve high performance in various environments and tackle challenging problems in complex and demanding tasks. In order to give a compact and practical overview of their differences, we present comprehensive comparison figures and tables, produced by reported performances of the algorithms under two popular simulation platforms: the Atari Learning Environment and the MuJoCo physics simulation platform. We discuss the key differences of the various kinds of algorithms, indicate their potential and limitations, as well as provide insights to researchers regarding future directions of the field.

1. Introduction

Deep Learning is one of the most popular fields in Machine Learning research, and has allowed scientists to deal with complex problems in a wide range of domains (Chalapathy & Chawla, 2019; Dargan et al., 2019). Reinforcement Learning (RL) is built upon the principles of learning via agent-environment interaction through actions and rewards (Sutton & Barto, 2018), which has opened up new pathways in the area of Artificial Intelligence (AI), achieving never-before-seen breakthroughs when combined with Deep Learning methods (Schrittwieser et al., 2019; Jaderberg et al., 2019; OpenAI, 2018; Silver et al., 2017; Vinyals et al., 2019). These encouraging results have motivated a wide spectrum of researchers to extend the boundaries of AI, and proceed towards the establishment of Artificial General

Intelligence (AGI) by applying Deep Learning methods in conjunction with Reinforcement Learning (Legg & Hutter, 2007; Geffner, 2018; Rocha et al., 2020).

A typical RL problem is mathematically formulated as a Markov Decision Process (MDP), and consists of an environment and an agent, who has the role of traversing the environment's states by executing actions and receiving a corresponding numerical feedback from it. The purpose of the agent is to maximize some kind of total cumulative reward until a termination condition is met (e.g. a state that flags the end of the interaction process). RL problems traditionally employ techniques that do not require the exact model of the MDP, which is also the main point of interest for this field, but there are also cases where methods for approximating this model are used. This is also the main categorization boundary for Deep RL methods, as we discuss later in detail.

Various works have investigated the charted areas of Deep RL methodologies. Francois-Lavet et al. (2018) and Li (2018) provide detailed guides on state-of-the-art Deep RL algorithms, evaluation platforms, research challenges and applications, but do not provide a novel restructure of algorithm categorization, or practical comparison information on performances between the algorithms. Similarly, Arulkumaran et al. (2017) developed a brief survey for Deep RL methods and algorithms that is also similar to our work, providing the basic theoretical background on RL formulation, introducing primary state-of-the-art Deep RL algorithms and discussing current research areas and challenges regarding this area, with a special focus on visual understanding. Even though such survey is of significant value, there is a non-trivial difference to the number of algorithms that are included in this work, as well as to the level of technical details for each algorithm. Additionally, we propose new ways for categorization of the algorithms, and also provide a novel and detailed collection of data and figures used for comparison purposes. The work of Shao et al. (2019) is in parallel to the style used in the aforementioned Deep RL surveys but differs in that the authors also include discussions regarding the achievements of Deep RL methods in video games. However, it lacks a more quantitative comparison of these methods, in contrast to our work. In other reviews, such as the ones by Da Silva and Costa (2019) and Nguyen et al. (2020), the authors focus solely in sub-categories of Deep RL methods, exploring in detail only a part of Deep RL and not an overview of the complete area.

Other existing surveys have explored the different applications of Deep RL, e.g. in healthcare (Yu et al., 2019), communications/networking (Luong et al., 2019), video games (Justesen et al., 2019; Skinner & Walmsley, 2019), cyber security (Nguyen & Reddi, 2019), natural language processing (Whiteson, 2019; Stylianou & Vlahavas, 2019), robotics (Tai et al., 2016), biology (Mahmud et al., 2018), urban sustainability/transportation (Nosratabadi et al., 2020), agriculture (Yang & Sun, 2019), and many others (Li, 2019).

All in all, there has been no organized attempt to gather state-of-the-art Deep RL algorithms in general, present and compare their unique and shared properties in a novel manner, as well as collect their reported performances in the most popular evaluation platforms, in one place.

In this review, we perform an analysis of several state-of-the-art Deep RL algorithms and divide them into three main categories, according to their anatomy and core functionality. The first presented category includes model-free algorithms, i.e. algorithms that do not rely on learning the state transition probability function for solving the optimization task in hand. In contrast, the second category consists of model-based algorithms, which tries

to approximate model dynamics in order to reach a solution. Each of these two categories contain various kinds of methodologies for the implementation of algorithms.

More specifically, model-free algorithms are distinguished in two commonly referenced subcategories: Deep Q-Learning and Policy Gradient approaches, both of which share a common structure. On the other hand, we classify model-based algorithms in a novel way, depending on whether the implemented methodology constructs latent states in order to make predictions, or uses only manifested (i.e. directly observable) states.

Apart from these two families of algorithms and their corresponding subcategories, we also explore the rest of existing Deep RL models in a unique way, presented in the category of modular algorithms. In this family, we give special focus on dissecting the model in two elements: the *core Deep RL algorithm*, which provides the essential functionality of the model, and the *framework* that hosts the core algorithm. This leads to a unique reporting of combinations between core algorithms and frameworks that have been implemented, as well as suggests potential combinations that have not been attempted yet.

This category is based on research performed on algorithms that are used as modules for functional Deep RL algorithms (such as the ones that fall in the two previous categories), developing robust architectures that improve original performance, and increase implementation flexibility and stability. Such components, for example, aim at improving exploration strategies, generalizing (transfer RL, multitasking), creating abstraction layers to solve sub-problems (hierarchical RL), learning skills without supervision (unsupervised RL), as well as developing distributed algorithms (distributed RL).

In this paper, we give a brief description and perform a compact comparative analysis of the state-of-the-art algorithms for each of these categories. Then, we present higher-level comparison figures of their performance on 57 games included in the Arcade Learning Environment (ALE) (Bellemare et al., 2013) and the MuJoCo physics simulation platform (Todorov et al., 2012), where applicable, as they were originally given by the corresponding authors. The reason for choosing these two platforms is essential to this review, since they have become the gold standard for testing Deep RL methodologies. These platforms would therefore allow for a wider and more accurate comparison of such algorithms, in terms of evaluation metrics, which is necessary for this analysis.

Primarily, this comparative survey is to be used as a guide for becoming acquainted with state-of-the-art Deep RL algorithms, knowing their pros and cons, their relations, performance capabilities, as well as distinguishing the cases where it's more appropriate for particular methods to be used. Such work allows the Deep RL community and researchers in general (especially early-stage researchers) to view specific traits of state-of-the-art algorithms in a categorized manner, as well as distinguish particular architectural designs developed for the means of advancing the fields of Deep Learning and Deep RL. Additionally, this kind of survey aims at giving insights regarding novel algorithm implementations and variations of existing algorithms. Lastly, we believe that the reader can develop intuition regarding capabilities and potential of the algorithms and their particular characteristics, under different evaluation scenarios.

In Section 2 we introduce the first family of algorithms, model-free algorithms.

2. Model-free Algorithms

Algorithms with a model-free nature make up the first major type of Deep RL algorithms, and constitute the epitome of a direct learning process through experience. More specifically, an agent within an environment attempts to learn the optimal policy for solving a task by directly transforming the experience gathered as a result of performed actions, into a resulting policy. In the following subsections, we describe two main categories of the model-free family, namely Deep Q-Learning and Policy Gradient algorithms.

2.1 Deep Q-Learning

Deep Q-Learning is based on Q-Learning approaches (Watkins, 1989), and has the purpose of approximating the optimal action-value function $Q^*(s, a)$ through the use of deep neural networks. To that end, the action-value function is parametrized as $Q^*(s, a; \theta)$, where θ is a vector of parameters on which the action-values rely to obtain their values, and is also the main target of approximation by the networks (known as Q-Networks), formally described in the following way:

$$Y_k^Q = r + \gamma \max_{a' \in A} Q(s', a'; \theta_k) \tag{1}$$

Deep Q-Networks (DQN) (Mnih et al., 2015) is the basis for all Deep Q-Learning approaches, motivated by the fact that it is impractical to use the traditional tabular Q-Learning method in environments consisting of large state spaces with high dimensions. However, DQN methods are limited to supporting discrete action and state spaces.

DQN showed notable performance on a wide range of Atari games, and partially owes this achievement to functionalities implemented by the authors that further increased stability in several cases where the system would diverge due to the use of a non-linear function approximator (Bhatnagar et al., 2009a). These modifications include using a separate Q-Network (called a target Q-Network) apart from the original (online) Q-Network. The target network is actually a clone of the original that is updated periodically to match the online network, which, on the other hand, is updated at every step. This is to maintain for a longer period the same action-values, in contrast to weight updates, which occur at every step. The lack of a target Q-Network introduces a high chance of divergence of the policy (Mnih et al., 2015). Additionally, the Experience Replay (ER) (Lin, 1992) technique is implemented in order to perform action-value updates from mini-batches of stored experiences, consequently increasing performance on a broad range of states. At the same time, variance of the updates is reduced since experience samples are drawn at random, which reduces correlation between samples, as opposed to single experience-tuple updates (Mnih et al., 2015). According to the authors' experiments, even though both modifications improve significantly model performance, the use of ER boosted results greatly, reaching in some cases about 30-70x higher performance scores in the ALE platform. An extra modification in this work was clipping rewards to the $[-1, 1]$ range so as to limit derivative error scale and use the same learning rate across different games, at the expense of introducing bias (Francois-Lavet et al., 2018).

Hasselt (2010) proposed Double Q-Learning, which, when combined with DQN (DDQN) (Van Hasselt et al., 2016b), improves performance by removing overestimation bias intro-

duced that would occur in some cases, because the same network would both evaluate and select an action. This is achieved by separating these two functions and using two independent estimators for each: the online Q-Network, which uses a set of weight vectors for selecting an action for the next state, and the target Q-Network, which uses a different set of weight vectors for the evaluation of the Q-value of that action. In particular, separation of action selection and evaluation is performed by decoupling the *max* operation in the original update rule (eq. 1). Then, the online network can select an action and the target network can evaluate that action, modifying the update rule as follows:

$$Y_k^{DDQN} = r + \gamma Q(s', \underset{a \in A}{\operatorname{argmax}}\, Q(s', a'; \theta_k); \theta_k^-) \tag{2}$$

Where θ_k and θ_k^- are the weights of the online and target networks, respectively. Human-normalized median of DDQN performance is 115% and 111% for the no-ops and human-starts schemes respectively (see Section 5.1) in the ALE platform environments, in contrast to 79% and 68% in DQN respectively (Hessel et al., 2018).

Another extension of DQN, Dueling DQN (Wang et al., 2016), is an architecture that splits the network into two streams, each of which corresponds to the decoupled Q-action value function, that is, the value function $V(s_t)$ and advantage function $A_{\pi_\theta}(s_t, a_t)$, through their relationship $A_{\pi_\theta}(s_t, a_t) = Q_{\pi_\theta}(s_t, a_t) - V(s_t)$. More specifically, instead of following the original DQN network architecture, a modification is implemented at the point where the convolutional neural network sends the output to a single sequence of fully connected layers. Instead of this structure, the output signal is sent to two different streams, with one being responsible for estimating the value function, and the other for estimating the advantage function. The outputs of these estimators are combined in order to produce a more accurate action-value function. The authors of this method found that the Dueling Network architecture on top of DDQN has a human-normalized median of 151% and 117% in the ALE platform games (Hessel et al., 2018), showing significant increase over both DQN and DDQN scores.

A variation of Dueling DQN, called Prioritized Dueling DQN (Wang et al., 2016), further improves performance of the original algorithm by incorporating the concept of Prioritized Experience Replay (PER) (Schaul et al., 2015b). PER appends a priority to each experience (i.e. transition) that corresponds to the Temporal Difference (TD) error from the last time it encountered that experience, with the purpose of sampling experiences with a larger TD-error more often during experience replay. However, this extension alone introduces high bias towards high-priority samples. To patch this issue, Importance Sampling (IS) weights (Hinton, 2007; Mahmood et al., 2014) are used to limit the effect of those priority experiences during updates. However, the practical impact of PER in overall model performance is not clearly visible, since the human-normalized median for the human-starts regime (128%) of Prioritized Dueling DDQN is slightly higher than Dueling DDQN (117%), but in the no-ops scheme it is slightly lower (140% vs. 151%) (Hessel et al., 2018).

DQN can also be modified to use Multi-Step Temporal Difference methods (Sutton, 1988; Hessel et al., 2018; Hernandez-Garcia & Sutton, 2019), which means that the loss to be minimized replaces the original one-step return from a state with an N-step return. This implies that bootstrap is performed over longer time periods, in contrast to one-step returns, which perform an action at the next step and bootstrap directly after receiving a

reward. While multi-step TD methods increase variance, it is proven that bias is decreased (Jaakkola et al., 1994), and a fine-tuned value of the step parameter can hasten the learning process (Sutton & Barto, 2018).

Oftentimes, the environment is less likely to reward the agent immediately after an action, making it necessary to perform a sequence of many actions until a reward is received. This difficulty of the agent interacting with the environment and obtaining "insights" about it, with the purpose of efficient learning, is known as the credit assignment problem in RL (Minsky, 1961; Sutton & Barto, 2018). Noisy Nets (Fortunato et al., 2018) is one method that improves the exploration process, something that is crucial in environments with sparse rewards. Exploration with this method is performed by replacing the fully connected layers of the network with noisy layers, which is, in essence, a randomized value function (Osband et al., 2019, 2016) that learns to adjust the intensity of exploration over time. DQN combined with NoisyNets surpassed standard DQN in terms of median human-normalized scores when evaluated in the ALE platform environments, achieving 118% and 102% in the no-ops and human starts regimes respectively (Hessel et al., 2018).

Another variant of DQN approaches, is to use a distribution-based algorithm instead, such as C51, in which one estimates the distribution of the returns, instead of the values of the expected returns (Bellemare et al., 2017). In their work, the authors highlight theoretically and empirically the significance of modelling this value distribution, and showed that C51 outperformed previous DQN variants in most games within the ALE platform, thus becoming a state-of-the-art algorithm. In particular, C51 managed to perform equally good or better than the aforementioned methods, achieving 125% (human-starts) and 164% (no-ops) median human-normalized scores (Hessel et al., 2018).

Rainbow (Hessel et al., 2018), an algorithm that is able to achieve remarkable performance in the ALE platform, is a combination of the aforementioned extensions of DQN, namely DDQN, Dueling DQN, Prioritized Dueling DQN, along with Noisy Nets, distributional RL and Multi-Step TD features. The achieved results are contributed to the performance gains of each one of its components, reaching 223% (no-ops) and 153% (human-starts) median normalized scores (Hessel et al., 2018), but it should be noted that the authors of Rainbow state that the dueling network component plays a less significant role in the Rainbow system, in general.

2.2 Policy Gradient Methods

In general, algorithms which make use of the Policy Gradient Theorem (Sutton & Barto, 2018) belong to the family of Policy Gradient (PG) methods. The base algorithm that belongs to this category is Vanilla Policy Gradient (VPG), or REINFORCE (Williams, 1992). In REINFORCE, the target is to maximize the expected finite-horizon undiscounted return J with respect to policy π, which, in turn, depends on parameters θ, with the help of gradient ascent methods. Gradient of $J(\pi_\theta)$ is defined as:

$$\nabla J\left(\pi_\theta\right) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) Q_{\pi_\theta}(s_t, a_t)\right] \tag{3}$$

Where τ is a trajectory, i.e. an interaction sequence (states, actions and rewards) between the agent and the environment until a terminal state is reached (also called episode).

Q is the action-value function for the current policy. The gradient is taken with respect to the policy parameters θ.

In other words, the purpose is to optimize the policy directly, which has the added benefit of supporting continuous action and state spaces, in contrast to DQN methods. This is derived by producing a policy parameterization, which outputs the parameters of a particular probability distribution (typically the mean and standard deviation of a Gaussian distribution), instead of computing the probabilities for an infinite amount of actions. Therefore, the policy can be defined as the probability density function of the Gaussian distribution, with the mean μ and standard deviation σ depending on parameters θ (for a particular state):

$$\pi_\theta(a_t|s_t) = \frac{1}{\sigma_\theta(s_t)\sqrt{2\pi}} \exp\left(-\frac{(a_t - \mu_\theta(s_t))^2}{2\sigma_\theta(s_t)^2}\right) \tag{4}$$

It should be noted that π on the right-hand side of eq. 4 refers to the mathematical constant, and not the policy. This way, one can know how likely an action is. By tuning μ and σ through θ, the probability distribution is optimized.

Even though PG methods are unbiased, they suffer from high variance and low convergence (Marbach & Tsitsiklis, 2003; Peters & Schaal, 2006; Sehnke et al., 2010). To reduce variance, a state-dependent baseline value b is usually subtracted from Q_{π_θ}, which keeps bias unchanged. A common approach in this case is to set the value of b as the state-value function, and subsequently compute the advantage function in the gradient, but other baseline values (e.g. action-dependent) can be chosen instead (Williams, 1992; Greensmith et al., 2004; Wu et al., 2018; Tucker et al., 2018).

The other important extension of base Policy Gradient methods is Actor-Critic (Konda & Tsitsiklis, 2000). In this variation, in addition to updating parameters θ in order to optimize the expected return, the baseline value of the Policy Gradient method used also becomes dependent on a set of parameters that are updated at each step and help optimize its value. This approach helps policy updates by reducing gradient variance. This type of Policy Gradient method makes use of two models, the actor and the critic. The critic updates parameters w in order to make an approximation of $Q_{\pi_\theta;w}$, in the case of Q Actor-Critic, or $A_{\pi_\theta;w}$, in the case of Advantage Actor-Critic (A2C) (Bhatnagar et al., 2009b) and Asynchronous Advantage Actor-Critic (A3C) (Mnih et al., 2016). At the same time, the actor uses this information to update policy weights, and, consequently the current policy. This way, an Actor-Critic method could be seen as a way to bridge PG methods (i.e. the actor) with methods based on value function approximation (i.e. the critic). The critic approximates and updates the value function using samples, which is then used to update the actor's policy parameters in the direction of performance improvement, effectively allowing a PG method indirectly exploit the MDP structure (Grondman et al., 2012).

In A3C, the implementation of which is based on parallelization, multiple instances of the agent are initiated in multiple instances of the environment, with each agent-environment setup being independent from the rest. The experiences gathered are sent asynchronously as input to a global network that updates the value function. A3C has considerably faster learning speeds than other methods, such as DQN, which the authors trained on a single Nvidia K40 GPU for 8 days only to achieve human-normalized mean and median of 121.9%

and 47.5%, respectively, using the human-starts evaluation scheme. A3C, on the other hand, was trained on 16 CPU cores, and achieved significantly better results (623% mean, 112,6% median) in half the time (4 days) (Mnih et al., 2016). A2C is the synchronous version of A3C, where trajectories created by a single agent in multiple instances of the environment are gathered and used as input to the global network. A2C can be more cost-effective than A3C on single-GPU machines, and is faster than a CPU-only A3C implementation when using larger policies (Wu et al., 2017b).

REINFORCE and its extensions (A2C, A3C, etc.), while providing a simple way of optimizing the expected return, are prone to making disastrous moves in cases where the objective function has low gradient (i.e. high curvature), due to the fact that it uses first-order derivatives. In that case, using small step sizes is inefficient due to slow learning (i.e. vanishing gradients), while large step sizes can hurt performance badly (i.e. exploding gradients) (Hochreiter, 2001). For this reason, Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) uses the Kullback-Leibler (KL) measure of divergence (Kullback & Leibler, 1951; Kullback, 1959) in order to constrain policy changes below a certain threshold at each iteration, subsequently avoiding destructive parameter updates. In essence, the KL-divergence measures the difference between two probability distributions, and in this case, the difference between a new policy and an old policy is measured. At the same time, monotonic improvement is guaranteed (Schulman et al., 2015). While TRPO addresses critical issues of Policy Gradient methods, it suffers from a crucial problem, which is the additional computational overhead required in order to satisfy the KL constraint, and also needs huge rollout batches to approximate the required conjugate gradient.

An improvement over TRPO that is less computationally expensive, and scales better in larger deep networks, is Actor Critic using Kronecker-Factored Trust Region (ACKTR) (Wu et al., 2017a). ACKTR is a Policy Gradient method that combines trust region optimization and the actor-critic architecture, with an extra modification that uses Kronecker-Factored approximation (K-FAC), which reduces computational complexity and delivers almost first-order optimization complexity (Martens & Grosse, 2015). In addition, it is able to reduce variance by keeping running averages of curvature statistics throughout training, consequently requiring far fewer samples, unlike TRPO. In particular, ACKTR managed to reach a reward of 2 million in the Atari game "Atlantis" in 1.3 hours (600 episodes), while A2C reached the same performance level in 10 hours (6000 episodes), which is a clear indication that ACKTR can be far more sample-efficient than other state-of-the-art algorithms. In other Atari games, A2C required 2-5 times more episodes than ACKTR to reach human performance level, while ACKTR achieved 25-70% larger rewards than A2C (Wu et al., 2017b).

Actor Critic with Experience Replay (ACER) (Wang et al., 2017) can be seen as the off-policy counterpart of A3C due to the fact that it uses experience replay, with a few extra modifications and additions that make it more stable and efficient. More specifically, they include estimating the action-value function using the Retrace algorithm (Munos et al., 2016), truncating importance weights with bias correction, as well as making use of trust region optimization by keeping a running average of past policies and constraining new policies to be near this average.

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a subcategory of Policy Gradient methods that maximizes gradient step size, without letting it become regrettably

big, similarly to TRPO. Unlike TRPO though, PPO algorithms incorporate first-order methods and reduce computational expense while maintaining satisfying policy update sizes, as well as simplify the implementation procedure.

Two main variants of PPO can be distinguished: PPO-Clip (or Clipped Surrogate Objective) and PPO-Penalty (or, Adaptive KL Penalty Coefficient) (Schulman et al., 2017). In PPO-Clip, there is no KL-divergence constraint within the main objective to be minimized. Rather, clipping is performed in order to constrain the new policy from going far off from the old policy; in other words, the objective function is upper-bounded, subsequently constraining the new policy from deviating as well. PPO-Clip is usually performed in multiple steps of taking minibatch Stochastic Gradient Descent (SGD).

In the PPO-Penalty variation of PPO, the hard constraint used for the KL-divergence is replaced with a soft constraint that takes the form of a penalty when not satisfied. Even though PPO is widely used due to its competitive results in combination with the ease of implementation, it displays a non-exploratory behavior over the course of training because of its tendency to exploit found rewards, since each updated policy becomes less and less random (Wang et al., 2019), and can get stuck in local optima. Additionally, PPO is highly-dependent on hyperparameter selection, requiring several experimentation procedures for fine-tuning.

Policy gradients have a stochastic nature, since the sampled policy stochastically selects an action from a particular state. On the other hand, there is a family of algorithms based on Policy Gradient methods, called Deterministic Policy Gradient (DPG) methods (Silver et al., 2014), where an action is selected in a deterministic way.

The most basic implementation of a DPG algorithm is Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016), which combines an actor-critic model with a Deterministic Policy Gradient (DPG) method, in order to develop a PG method with efficient exploration in continuous action spaces. For this purpose, DDPG embodies two distinct functionalities that improve model performance and stability: the first is batch normalization (Ioffe & Szegedy, 2015), for normalizing values of features with different units, so as to maintain performance even when generalizing to environments with different state value scales. The second functionality aims to handle exploration insufficiency in continuous action spaces, by creating an exploration policy with added noise generated by an Ornstein-Uhlenbeck process (Uhlenbeck & Ornstein, 1930). Performance of DDPG is competitive to state-of-the-art approaches with increased sample-efficiency in high-dimensional continuous action spaces, solving such problems in 20x fewer steps than what a typical DQN model requires to solve Atari games, which are considered as environments with low-dimensional discrete action spaces. However, DDPG requires a lot of tuning for a satisfactory setup. Additionally, errors in the learned Q-function can easily lead to catastrophic divergence of the policy (Duan et al., 2016; Henderson et al., 2018).

Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) addresses the issues of DDPG with three modifications, improving both learning speed and overall performance: 1) Clipped Double Q-learning, for reducing overestimation bias, 2) Delayed policy updates for decreasing variance in value estimates by updating less often policy and target networks, and 3) Target policy smoothing regularization, which introduces noise to the target policy, averaged over mini-batches, but clipped to keep the new target close to the original, so as to further reduce variance.

A bridge between Q-learning and DPG methods is Soft Actor-Critic (SAC) (Haarnoja et al., 2018a), which introduces a maximum entropy term (Haarnoja et al., 2017; Jaynes, 2003; Ziebart et al., 2008) that corresponds to efficient exploration, and attempts to maximize it. Theoretical analysis behind this method guarantees convergence to optimal entropy policy. This maximum entropy framework is implemented in an off-policy fashion, and is combined with a stochastic actor-critic, with the end result empirically outperforming state-of-the-art on-policy and off-policy algorithms in terms of sample efficiency, as well as performance, in high-dimensional continuous control tasks. However, manual tuning of the temperature hyperparameter in the maximum entropy setting can degrade performance, a problem which is tackled by a modified version of SAC, introduced in (Haarnoja et al., 2018b), by automating the process of selecting the optimal temperature hyperparameter.

3. Model-based Algorithms

Model-free approaches rely on the experience acquired from interaction with the environment in order to develop a policy for solving the task in hand. However, this can be seen as a disadvantage, since complex domains usually require that the agent obtains a great deal of experience before efficient policies are learned. Additionally, even after the agent becomes a master at solving a task within an environment, the exploitation of existing knowledge for solving novel tasks in the same or a different environment (generalization) is not an implied ability. Model-based learning methods try to address these flaws by attempting to learn the underlying environment stochastic dynamics using the experience gained, so as to accurately predict future states, and then use planning approaches to solve the given task (as in a purely dynamic programming approach). In contrast, a model-free agent tries to learn and execute the best action for a given state (i.e. learn the optimal policy).

In model-based learning, the agent uses past experience to create an imagined model of the environment (i.e. transition probabilities and rewards), and thus predict the outcomes of performed actions (i.e. future states), subsequently avoiding negative actions and potentially destructive pathways. This also allows for better generalization across different tasks within the same environment, since there is a lesser need for large amounts of real experience in order to solve them (Deisenroth et al., 2013). The main obstacles within this category of Deep RL methods are usually model inaccuracies, accumulating errors of multi-step predictions, failure to capture multiple possible future states, and overconfident predictions outside of the training distribution (Hafner et al., 2019).

Two basic categories of algorithms are distinguished in this family, depending on whether the algorithm uses techniques for finding latent states (non-directly observable) in order to create the model of the environment, or if only manifested (directly observable) states are used. Manifest state-based and latent state-based algorithms are described in the following subsections.

3.1 Manifest State-Based Algorithms

Probabilistic Inference for Learning Control (PILCO) (Deisenroth & Rasmussen, 2011) is a model-based policy search algorithm that models the required world dynamics using non-parametric Gaussian Processes, uses approximate inference for policy evaluation, and computes analytic derivatives with respect to optimization objective function parameters

for policy improvement. It is highly data-efficient due to its probabilistic nature of learning dynamics models; for example, it learned the dynamics of a real Cartpole system in less than 10 trials that correspond to total experience of 17.5s, whereas other similar methods required one or more orders of magnitude more time to achieve the same. However, PILCO struggles in high-dimensional state spaces and nonlinear dynamics (Nagabandi et al., 2018), while at the same time has high computational requirements.

Another widely used hybrid approach is Model-Based Model-Free (MBMF) (Nagabandi et al., 2018), the design of which consolidates a model-based methodology that uses Model Predictive Control (MPC) with Random Shooting (RS) (Richards, 2005) to initially train a system for learning model dynamics. Once this model-based RL component has been trained, sampled trajectories are used as "expert" trajectories for the DAGGER imitation learning algorithm (Ross et al., 2011), in order to produce a corresponding policy. This policy is used as the initial policy for a model-free RL agent (the authors originally use TRPO), which is then trained in a straightforward fashion. Although the model-based component of the algorithm is not capable of reaching high rewards, it is able to quickly provide significant insights about the model dynamics, so that the model-free approach can then use this information to achieve high performance, with sample-efficiency of about 3-5 times compared to pure TRPO in MuJoCo tasks. However, the MPC controller with RS is bound to have trouble in environments with high-dimensional action spaces, in which sample efficiency would drop significantly.

Recent research has also dealt with ensemble approaches for solving Deep RL tasks, such as Stochastic Ensemble Value Expansion (STEVE) (Buckman et al., 2018). The main contribution of STEVE is the extension of Model-Based Value Expansion (MVE) (Feinberg et al., 2018) for use in an ensemble setup, in combination with a model-free method (originally DDPG). Performance experiments in continuous control tasks show that STEVE outperforms MVE-DDPG and DDPG in terms of both sample efficiency and wall-clock time, in fact achieving an order of magnitude increase in sample efficiency within MuJoCo environments, compared to DDPG. Similarly, Model-Ensemble Trust-Region Policy Optimization (ME-TRPO) (Kurutach et al., 2018) uses an ensemble of deep neural networks in order to manage uncertainty, and performs policy optimization using a model-free algorithm (originally, TRPO was used by the authors). However, ME-TRPO is a purely model-based technique, in contrast to STEVE, which alternates between model-based and model-free methods throughout execution. The non-trivial sample efficiency of ME-TRPO can be clearly seen from its performance in MuJoCo tasks, where it reaches the same performance level as model-free algorithms with 100x less data.

One particularly promising direction was pointed to by the authors of Model-Based Meta-Policy-Optimization (MB-MPO) algorithm (Clavera et al., 2018), in which meta-learning is used to optimize a policy for each (environment) dynamics model within an ensemble. Each dynamics model corresponds to a different task that the agent shall be trained on, combating model bias with adaptation to different scenarios, rather than exercising robustness. Policy parameters are optimized with Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017), while the objective of the meta-optimization problem is optimized using a model-free technique (originally, the authors used TRPO). Sample efficiency of this method in comparison to model-free approaches is evident in MuJoCo tasks, and its increase ranges from 10-100x, depending on the task.

Chua et al. (2018) propose using high-capacity systems robust to uncertainty generated by the environment, and develop Probabilistic Ensembles for Trajectory Sampling (PETS), a sample-efficient and powerful algorithm composed by uncertainty-aware deep networks. An ensemble of probabilistic networks tune the parameters of a probability distribution function, which is the core of the agent's model-learning method. For the navigation procedure within the environment, the agent makes use of an MPC controller with Trajectory Sampling. The authors' experimental results show significant improvement of both state-of-the-art model-based and model-free algorithms; in particular, PETS requires 8x less data than SAC and 125x less data than PPO on the Half-Cheetah MuJoCo task, to reach equal performance.

3.2 Latent State-Based Algorithms

Deep Planning Network (PlaNet) (Hafner et al., 2019) is a model-based reinforcement learning algorithm that identifies latent states between real states during transitions, in order to represent accurately transition dynamics. Raw image pixels are the only input to the model, and various components make up the whole algorithm. Sample efficiency in this case is increased by 200x on average in continuous control tasks, compared to state-of-the-art model-free algorithms such as A3C and D4PG, with final performance sometimes also being higher.

More specifically, it uses the Cross-Entropy Method (CEM) (Chua et al., 2018; Rubinstein, 1997) for the planning component due to its performance on solving the required tasks under the condition that the true dynamics of the model are known. Additionally, due to the fact that a deterministic transition model design would prove to be inaccurate, while a purely stochastic model would have difficulties remembering information over long past time periods, the authors used a Recurrent State-Space Model (RSSM), in which a state is split into a deterministic and a stochastic part, with the former used for remembering past information over multiple steps, and the latter used for predicting multiple future scenarios. The last component of the PlaNet algorithm is latent overshooting, a generalized variational bound that provides the agent with the ability to perform multi-step predictions, instead of only a single-step prediction. Performance results indicate that working with latent state spaces is a promising direction for model-based RL methods.

A hybrid approach, which combines model-based elements with a model-free algorithm, is Imagination-Augmented Agents (I2A) (Racanière et al., 2017). This particular method incorporates an imagination core module used for producing possible future trajectories (i.e. rollouts) from past experience and through action-conditional next-step predictors (Chiappa et al., 2017; Leibfried et al., 2017; Oh et al., 2015). Imagined rollouts are then encoded using LSTM encoders (Hochreiter & Schmidhuber, 1997) with the purpose of extracting valuable information, as in making interpretations, in order to reduce model bias, that is, ignore the assumption of a perfect learned model (Deisenroth & Rasmussen, 2011). At the same time, a model-free agent (the authors originally use an A3C agent) is trained naturally at each time step, only to feed both their output and the concatenated model-based encoded rollouts into a policy network which undertakes the task of producing the final action to be executed. The key advantage of this method is its ability to distinguish useful information from the learned models, and subsequently noisy environments. The authors evaluated

their model in Sokoban, a puzzle game with 40 billion procedurally generated levels, as well as MiniPacman. I2A which solved 85% of the Sokoban puzzles, with other baseline model-free methods solving only 60-70% of them at potentially less cost. The MiniPacman environment was used to evaluate generalization capabilities of the model, which learned to perform efficiently (i.e. achieve high rewards) all 5 different tasks within the environment, but other baseline models could not learn more than 1 task at a time.

World Models (Ha & Schmidhuber, 2018) incorporates a Variational Autoencoder (VAE) (Kingma & Welling, 2013) to produce an embedded representation of the observed input image, which is then passed to a memory module. The memory module has the role of predicting the next latent state, by training a Mixture Density Network (MDN) combined with a Recurrent Neural Network (MDN-RNN) (Graves, 2013) in order to model the probability distribution of the next latent state, while introducing randomness within this process in order to better handle model uncertainty (Ha & Eck, 2018). Finally, a controller module, that is, a single layer linear model, is trained to output an action given a latent state and a hidden unit of the RNN. Due to the small size of parameters of the controller, the authors chose to use Covariance-Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen & Ostermeier, 2001; Hansen, 2016) as the optimization algorithm.

This algorithm has the benefit of requiring a relatively small sample of real observations for generating its own perception of the environment and training in it, plus requires lower computational resources due to the conversion of image input to embedded representations. This is evident by its performance in the CarRacing-v0 (Klimov, 2016) and Viz-Doom (Kempka et al., 2016) environments, where it learns to reach the solutions and achieves state-of-the-art scores compared to model-free methods such as DQN and A3C with relatively few real samples of data. However, it is limited by nature to learning representations that are either relevant only to low-complexity tasks, or completely irrelevant to any of the necessary tasks. This is because the number of useful representations that can be captured is limited by the number of significant observations that have been received, and subsequently by the complexity of the environment and dynamics.

Simulated Policy Learning (SimPLe) (Kaiser et al., 2019) is a hybrid algorithm that provides stochasticity-instigating techniques to handle uncertainty, such as the introduction of discrete latent variables (Kaiser & Bengio, 2018) and scheduled sampling (Bengio et al., 2015; Venkatraman et al., 2016). This model achieves satisfying performance on a wide variety of Atari games (originally implemented with PPO as the model-free component), being 10x more sample-efficient, since it requires only 100K steps to reach the same performance that Rainbow and PPO reach at 1 million steps in Atari games. This performance in the ALE platform can place SimPLe into an overall top position among state-of-the-art Deep RL algorithms.

4. Modular Algorithms

This family of algorithms is composed of frameworks designed to host model-free or model-based algorithms, such as those mentioned in the previous chapters. These frameworks aim to improve the injected algorithms in particular aspects, as in improving the strategy used for exploration, and/or equip them with various new features, such as the ability to work in a distributed fashion, transfer knowledge, multitask, distinguish temporal abstractions

(hierarchical RL), or learn without the help of a reward function (unsupervised RL). These are the cases in which a state-of-the-art algorithm can exceed its own limitations and become a top-ranking algorithm in otherwise challenging tasks.

An analysis of Deep RL algorithms that belong to subcategories of the modular algorithms family (distributed frameworks, exploration frameworks, unsupervised frameworks, hierarchical frameworks and generalization frameworks) are provided in the following subsections.

4.1 Distributed Frameworks

Distributed frameworks aim to improve performance by scaling up existing algorithms. Ape-X (Horgan et al., 2018) achieves this by creating a parallelized version of PER, through the use of multiple actors, each of which interacts with its own instance of the environment. Generated data is added to a single memory container, which is accessed by a single learner component in order to sample from it, apply the implemented learning rule and update sampled experience priorities. Actors' network parameters are updated periodically (so as to stabilize learning, as proposed by Van Hasselt et al., 2016b), while old experience is also removed periodically, to save memory space. Originally, the authors applied the described framework to a variant of DQN (termed Ape-X DQN), which is similar to Rainbow (only double Q-learning, multi-step TD and dueling network architecture components are used instead of all Rainbow components), as well as to DDPG (termed Ape-X DPG). Results indicate that this algorithm outperforms standard state-of-the-art baselines, both in performance and wall-clock time, with Ape-X DQN achieving a human-normalized median of 434% (no-ops) and 358% (human starts) in the ALE platform, however this algorithm is mainly suitable for use only on occasions where large amounts of data can be generated in a parallel manner.

A multifaceted framework for the overall improvement in performance of a core model-free algorithm is Distributed Distributional Deep Deterministic Policy Gradient (D4PG) (Barth-Maron et al., 2018) (originally, the authors use DDPG as the core RL algorithm). The main characteristic of D4PG is its distributional nature, inspired by the work of Bellemare et al. (2017), which, along with N-step returns, demonstrate a strong behavior. Additionally, it has a distributed architecture that is based on Ape-X, with the authors claiming that, although PER can lead to performance improvement, it plays a less important role than the previous two features, and can also lead to unstable results. These two points are partially in line with the conclusions of the authors of Ape-X, since their claim is that applying PER does not hurt their algorithm neither in terms of performance or stability.

In the same style as Ape-X and D4PG is Importance Weighted Actor-Learner Architecture (IMPALA) (Espeholt et al., 2018), which is based on the Actor-Critic method. IMPALA, apart from the use of multiple actors and a centralized learner like Ape-X and D4PG (although it also supports a central set of multiple learners communicating with each other), does not obligate actors to compute gradients themselves and let the learner(s) apply updates. Instead, actors have a very specific role of sharing only observations with the learner(s), who, in turn, compute gradients themselves and, in the case of multiple learners, perform updates synchronously. This creates a practical independence between the two components (learners and actors), since actors do not have to wait for the learners to ap-

ply updates, and learners can perform updates without consuming computational resources that are used by the actors.

However, this also develops a policy-lag, since the actor's policy is several updates behind the learner's policy, a problem addressed by the authors through the use of a modified version of the Retrace algorithm, which they term V-Trace. V-Trace corrects the state-value function difference caused by the lag, in contrast to Retrace, which corrects the action-state value function instead. IMPALA, as the authors' experimental results indicate, manages to process humongous amounts of data in relatively short times at the cost of computational power, while also being able to cope with multitask settings and achieving satisfactory performance on a wide variety of environments. For example, IMPALA with 1 learner, it can reach in 10 hours the same performance that A3C reaches in 7.5 days, whilst a higher number of actors speeds up the learning process even more. Additionally, its generalization abilities are evident from the fact that a single model, with a single set of weights, managed to learn all games within the ALE platform, achieving a human-normalized median of 59.7% (no-ops) but at a cost of 11.4 billion frames in total.

A different approach to distributed RL is the General Reinforcement Learning Architecture (Gorila) (Nair et al., 2015), which promotes the use of 4 different distributed elements in its structure: multiple actors and multiple learners that work in parallel, a distributed experience replay memory (either local, with each actor storing his experience in his own machine, or global, where experience is inserted into a distributed database and accessed by one or more learners), as well as a distributed neural network maintained by a central parameter server, as in DistBelief (Dean et al., 2012). Originally, the authors supported the implementation of DQN with this framework (termed Gorila DQN) with local replay memory (whereas global replay memory would bring the framework one step closer to the architecture of Ape-X). Even though ALE benchmark results show clear improvements over DQN and other algorithms, both in overall scores and time required to achieve them (e.g. it surpassed performance of DQN on 19 Atari in 6 hours, and on 38 games in 36 hours), it requires a lot of computational resources, and is also prone to instability in case a single machine fails.

4.2 Exploration Frameworks

Efficient exploration is one of the most significant challenges in RL, and any improvement to this problem can lead to the overall increase in performance of any RL algorithm. A well-known basis for exploration methods is curiosity-driven exploration, which arises from the concepts of curiosity (Silvia, 2012) and intrinsic motivation (Oudeyer & Kaplan, 2009; Schmidhuber, 2010). These notions describe one's compulsion to explore an unknown part of an environment, with the belief that this shall lead to higher rewards at some point in the future.

In curiosity-driven exploration (Pathak et al., 2017), the authors propose the Intrinsic Curiosity Module (ICM), which attempts to create an intrinsic reward signal from useful feature encodings of the state space, that, in essence, is the prediction error of feature encodings from consequent states (i.e. the uncertainty of the environment). This model overcomes the problem of dealing with complex visual input (that is, pixels), and pays attention only to useful features that can affect the agent substantially. Consequently, the

agent uses the prediction error for these features as a measure of curiosity and motivation for exploration (i.e. higher error indicates higher curiosity and thus more interest for exploration). Originally, the authors implemented ICM with A3C as the framework's core RL algorithm. A good example of its exploratory abilities is that the agent can complete more than 30% of Level-1 in the Super Mario Bros. game, with absolutely no extrinsic rewards. However, it should be noted that a sequence of more complex actions is required to surpass that point where the agent is unable to proceed further into the game, indicating that the proposed curiosity methodology is indeed remarkable and also has interesting research potential.

The Variational Information Maximizing Exploration (VIME) (Houthooft et al., 2016) framework is also a curiosity-driven model, which maximizes information gain (Cover, 1999; MacKay, 1992) using variational inference (Jordan et al., 1999; Wainwright & Jordan, 2008) to approximate the posterior distribution of a Bayesian neural network (Blundell et al., 2015; Graves, 2011) that expresses environment dynamics and subsequently minimizes uncertainty. This model is used as the intrinsic reward function and is combined with the environment's extrinsic reward function, while a single hyperparameter in the model can be adjusted to control the exploration-exploitation trade-off. VIME can be used in conjunction with any standard RL algorithm, as long as it supports continuous state and action spaces (originally, VIME was implemented on top of TRPO). Authors' experiments prove that the model's exploratory abilities allow it to perform better than other methods (e.g. TRPO) in environments with sparse rewards such as Mountain Car (Moore, 1991).

Count-Based Exploration methods, on the other hand, support the idea of counting state visits as a base for exploration. However, to count directly state visits in an environment with a large state space is a challenge that cannot be bypassed easily, since, realistically, most states will not be visited more than one or two times (Burda et al., 2019).

In order to overcome this problem, the measurement of a pseudo-count quantity can be used, as in Context Tree Switching (CTS)-based pseudocounts (also known as A3C+) (Bellemare et al., 2016, 2014) and PixelCNN-based pseudocounts (Ostrovski et al., 2017; Van Den Oord et al., 2016), where the authors extend information gain-based algorithms such as VIME. This pseudo-count quantity is related to (or, better, constrained by) the improvement in prediction of a CTS and PixelCNN density model over the state space respectively, and used within a Model-Based Interval Estimation with Exploration Bonuses (MBIE-EB) setting (Strehl & Littman, 2008) that adjusts the exploration level in proportion to information gain. This method helps exploration by generalizing visit count over states. Originally, the authors implement A3C for use within CTS-based pseudocounts exploration framework, and DQN for PixelCNN-based pseudocounts. It is noteworthy that all three methods perform descently on the Atari game Montezuma's Revenge, showing interesting exploratory skills, where most state-of-the-art Deep RL methods do not achieve a score higher than zero.

Similarly, in hash-based counts (Tang et al., 2017), the authors propose hashing the state space into a bit-string of a predefined constant length, with hash collisions representing state revisits and/or similar states, then encouraging exploration in states indicated as novel. While the authors use locality sensitive hashing (LSH) (Bloom, 1970), and more specifically, SimHash (Charikar, 2002) for generating hash codes, they also suggest that using an autoencoder to learn the hash codes improves the method's accuracy. Originally, the

conducted experiments indicate significant performance on challenging continuous control tasks and Atari games, using TRPO as the core algorithm.

A similar concept to count-based exploration methods that produce density models is proposed in the EX2 algorithm (Fu et al., 2017), the core of which is based in exemplar models (Malisiewicz et al., 2011). In EX2, a single amortized model is trained to use an exemplar model as input (i.e. multiple classifiers trained to distinguish novel states from old states) and produce state densities, only to perform novelty-based exploration using the output from the exemplar models as exploration bonuses. As in most previous cases in exploration frameworks, the authors originally use TRPO for measuring performance. EX2 appears to be slightly better than other state-of-the-art methods, but a wider range of experiments and comparisons should be performed in order to reach more accurate conclusions.

Random Network Distillation (RND) (Burda et al., 2019) approaches this matter slightly differently, since it uses a fixed and randomly initialized target neural network that feeds on raw pixel input and generates, to train a predictor neural network on the agent's collected data by minimizing the expected Mean Square Error (MSE) between them, only to use this prediction error as an intrinsic, exploration reward, similar to an exploration bonus. By intuition, this distillation process gives the predictor network the ability to distinguish novel states, since in these less familiar cases the error is presumed to be higher, urging the agent to explore them until the error decreases (i.e. these states have been explored and are not new anymore). The efficiency of this method is backed empirically by the results of the authors' conducted experiments, with PPO used as the core RL algorithm for the framework. These experiments included hard-exploration Atari games, in which the model resulted in high performance, and the highest score that we managed to record, in Montezuma's Revenge.

A unique method for efficient exploration is Deep Exploration via Randomized Value Functions (Osband et al., 2019). Deep exploration refers to the ability of an agent to know how an action will lead to improved and more efficient information gain in the future, and not just in the subsequent timestep. A detailed theoretical background on this method is given, but the basic concept is that sampling is performed from a proxy of the posterior distribution over value functions, and actions performed must be greedy under this randomly drawn value function, which can be viewed as an extension of Thompson sampling. The Randomized Least-Squares Value Iteration (RLSVI) (Wen, 2014) class of algorithms is used for applying Deep Exploration and evaluating exploration efficiency in a set of problems, and a comparison with the exploration capabilities of DQN in a modified version the Cartpole environment is also performed. The results indicate the superiority of Deep Exploration via Randomized Value Functions over DQN with regard to exploration, since DQN only managed to receive a reward of 0, while the proposed method learned to solve the task relatively easy. This also shows the algorithm's potential if combined with Deep RL algorithms.

4.3 Unsupervised Frameworks

In RL problems where extrinsic rewards are nonexistent, it is up to the agent to craft an intrinsic reward function herself in order to cope with the task at hand. Problems of this

kind can be viewed as a subset of problems related to environments with sparse rewards, and even though the exploration methods described previously could be tuned for zeroing extrinsic rewards, approaches driven specifically by the lack of this information are geared towards learning multiple high-level skills (called options) within an environment, a process known as option discovery (Bacon et al., 2017; Precup, 2000; Sutton et al., 1999), rather than maximizing an objective function that corresponds to a single skill.

In this context, one can note many similarities between methodologies that tackle hierarchical RL as well. Against this background, notable examples of unsupervised RL algorithms are Variational Intrinsic Control (VIC) (Gregor et al., 2017) and Diversity Is All You Need (DIAYN) (Eysenbach et al., 2019), both of which employ information-theoretic approaches. VIC is based on the measure of empowerment (Klyubin et al., 2005; Salge et al., 2014) (originally used in conjunction with the Q-learning algorithm), which maximizes the mutual information between actions and future states, while DIAYN maximizes the mutual information between states and skills. This difference in DIAYN, as the authors state, can be interpreted as "maximizing the empowerment of a hierarchical agent whose action space is the set of skills". At the same time, DIAYN (which is originally implemented with SAC as its core RL algorithm) minimizes mutual information between actions and skills, which guarantees that learned skills are the result of states and not actions.

A key difference between VIC and DIAYN is that the latter avoids learning the prior distribution over skills. The authors argue that this process causes a crash in the number of skills sampled and consequently trained, which springs from what is generally referred to as the Matthew Effect (Merton, 1968).

Variational Autoencoding Learning of Options by Reinforcement (VALOR) (Achiam et al., 2018) is, in essence, a generalization of the previous two methods that encodes whole trajectories regularly instead, using an extension of VAE, namely β-VAE (Higgins et al., 2017). This generalization of VIC and DIAYN allows VALOR to be considered as an overall top choice among the three for unsupervised RL problems (excluding implementation details). For VALOR, the authors originally used VPG as the core RL algorithm for the benchmark tests.

4.4 Hierarchical Frameworks

Hierarchical RL could be described as an even more specific case of unsupervised RL, in which the exploration process within an environment with no rewards (although sparse rewards is also a setting suitable for hierarchical RL), is performed by generating intrinsic sub-goals and behaviors that solve them. The purpose of the methodologies within this category is to primarily guide the agent into learning skills that could prove to be useful for multiple tasks in the current environment (accounting for the possible lack of supervised rewards) as well as create a set of skills that can generalize to other environments.

h-DQN (Kulkarni et al., 2016) is a basic method with a relatively simple hierarchical structure that develops temporal abstractions through the use of a meta-controller (originally, DQN is used for the original implementation of the algorithm). FeUdal Networks (FuNs) (Vezhnevets et al., 2017) is a framework with a similar architecture as h-DQN but with noteworthy differences, and is inspired by Feudal Reinforcement Learning (Dayan & Hinton, 1992).

This system consists of two modules: a manager, who learns to set abstract goals from a lower temporal resolution in a latent state-space (which is also learned by the manager herself), without holding any information on how they can be achieved, and a worker, who learns to satisfy these goals in a higher temporal resolution. While the manager is rewarded by the environment, the worker is rewarded by the manager instead, through the use of an intrinsic reward function that is in essence a direction in the state-space. Originally, the authors implemented A3C as the reinforcement learning routine. Even though both h-DQN and FuNs try to learn particular skills within an environment and solve sub-goals, h-DQN requires only about 4 million steps in order to get consistently high rewards (i.e. 400 points) in Montezuma's Revenge, in contrast to FuNs, which need about 300 million steps to reach the same performance level.

Strategic Attentive Writer (STRAW) (Vezhnevets et al., 2016), on the other hand, generates macro-actions (McGovern et al., 1997) which are based on the agent's generated internal plans. STRAW also consists of two modules; one for following the current plan, and one for setting its termination conditions and updating it once stopped, using attentive writing technique (Gregor et al., 2015). With the implementation of A3C in its core, apart from achieving non-trivial performance on various Atari environments (but at the expense of hundreds of millions of frames), the authors show that STRAW can also be used for sequence prediction in general.

Universal Value Function Approximators (UVFA) (Schaul et al., 2015a) are also a primary method for hierarchical RL, which expands the notion of value functions to being dependent on goals as well, instead of relying solely on states. The original UVFA algorithm uses Q-Learning to perform updates.

A relatively unique algorithm is Hierarchical Reinforcement Learning with Off-policy Correction (HIRO) (Nachum et al., 2018), in that it uses raw observations instead of processed signals in order to aid generalization issues, and, more importantly, has an off-policy nature supported by a correction method for stabilizing communication between high-level controllers (guided by new policies), and low-level controllers (which generate data regarded as "old" when received by the higher-level modules). HIRO was originally implemented with DDPG as the core reinforcement learning method, and managed to outperform baseline models such as FuNs and VIME on MuJoCo tasks in a significantly more sample-efficient manner, requiring only about 10 million agent-environment interactions to achieve good performance.

In contrast to HIRO, the off-policy algorithm Hierarchical Actor-Critic (HAC) (Levy et al., 2019) uses a parallel training between all levels within the hierarchical architecture of the system, in a bidirectional fashion (all policies are trained concurrently), for speeding up the learning process. HAC allows the user to define a maximum number of steps (defined manually by the user) for achieving sub-goals, and applies Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) in each hierarchy level for learning from missed sub-goals. While this agent achieves decent performance on the environments tested by the authors when attached to the DDPG algorithm, notable comparison studies against a variety of baseline methods are not shown (apart from a comparison with HIRO, but with no explicit information regarding sample efficiency); even the resulting performance of the model with more than 4 levels of hierarchy is not presented either.

In Meta Learning Shared Hierarchies (MLSH) (Frans et al., 2018), the authors apply a meta-learning process in order to optimize several sub-policies in parallel, for various different but sequential tasks sampled from a distribution. The results provided originally are generally acceptable for a state-of-the-art algorithm (originally using PPO as its core RL algorithm) and appear to have small variance over many iterations, however it requires a decent amount of hyperparameter fine-tuning in order to achieve better performance than other algorithms. Nevertheless, this hierarchical type of learning succeeds in tasks where exploration over the space of actions is futile but exploring sub-policies is effective.

4.5 Generalization Frameworks

Generalization refers to the ability of an agent to function efficiently in new, unseen tasks and environments. There are various approaches for achieving this goal, e.g. through transfer knowledge or multitasking, which are two very similar concepts in RL. Fitting a single agent to multiple environments that share only a few properties in terms of structure (that is dynamics, action/state space or rewards) poses a challenge in that, even though an agent can be trained with a standard state-of-the-art Deep RL algorithm on a set of different environments with success, it is not able to generalize well after training on only one environment and adapt quickly to the rest.

Various frameworks have been developed for addressing the issue of task generalization. Hessel et al. (2019b) analyze various components that produce inductive bias within Deep RL algorithms. PopArt (Hessel et al., 2019a; Van Hasselt et al., 2016a) opposes the problem of one of these components (magnitudes of returns across different domains) and achieves impressive results under the aforementioned conditions, by creating an agent that performs scale-invariant updates, extended to the multitask setting. Only a single instance of an agent (the original version of which is implemented with the IMPALA extension for distributional RL), is to be trained consequently on a series of environments, and, as the authors' results indicate (and is also seen in Table 1 of the Appendix), it learns most Atari games, achieving a human-normalized median of 110%. Even though it does not absolutely outperform all other state-of-the-art algorithms, it has remarkable performance with the same amount of training frames as if it were independent agents trained individually on the different games.

A relatively different approach is proposed by Hausman et al. (2018), where the agent learns and executes different skills through a learned latent space, by using a variational bound on entropy-regularized RL, originally developed with an off-policy structure similar to DQN, but with additional tweaks such as incorporating a variant of Retrace algorithm and applying the reparametrization trick (Kingma & Welling, 2013; Rezende et al., 2014). The model was evaluated on a set of continuous control tasks and learned to solve them effectively in a sequential manner, however the baseline models used for comparison purposes do not include other similar state-of-the-art methods.

Cabi et al. (2017) propose the use of the Intentional Unintentional (IU) Agent, who leverages information gained by automatically-generated reward functions and trains off-policy multiple unintentional (submain) policies and a single (main) intentional policy. Even though the authors claim that the results support their intuitions that this learning process is effective, highly-custom environments are used for their experiments, something that does not allow much space for confidence that this method can be used to other, more complex

environments as well (e.g. environments with objects that do not have clear relational properties which the agent could easily exploit "subconsciously").

Probably one of the most eminent works for transfer learning in Deep RL, and deep learning in general, is PathNet (Fernando et al., 2017), which is based on the work by Fernando et al. (2011). The authors of PathNet propose using a meta-learning approach in order to speed up learning of new tasks, after learning a single task. The process could be described as "neural pathway evolution and optimization", since neural pathways are chosen as the result of an evolutionary method, while at the same time a learning process takes place in order to train the chosen paths (originally, the A3C algorithm was used). Moreover, after the optimal pathway has been found, its weights and biases can be held fixed, in order to be used at a different task, meaning that the pathway is kept frozen but the value function and policy are still being trained. This characteristic, inspired by Progressive Neural Networks (Rusu et al., 2016), serves as a way of avoiding catastrophic forgetting (French, 1994; Kirkpatrick et al., 2017; Ratcliff, 1990) when the new task is introduced.

Published experiments using PathNet are highly satisfactory, since they show empirically that performance is better with the proposed transfer learning methodology, compared to training on the different tasks de novo (even when they are fine-tuned). However, there is still space for expansion in PathNet, e.g. analyzing the network's memory capacity when a longer chain of tasks is to be learned (as in the work by Rusu et al., 2016), or testing for sample efficiency the evolutionary strategy used and exploring other variants.

5. Comparative Results

In this section, we present performance figures of the aforementioned algorithms, where applicable, as extracted from the original experiments conducted by the corresponding authors. These results are indicative of each algorithm's abilities, due to the fact that, at that point, they were all carefully fine-tuned so as to present considerable results. Even though this means that these methods can behave quite differently in new scenarios (and quite possibly worse than expected), one should also consider that the reported performances are indeed achievable, especially in the cases where extensive experimentation has been performed.

Unfortunately, comparing the original results under the same common framework is not an easy task, since not all results are produced from the same testing environments or under the same conditions. Although comparative figures for said methods are provided in this work, it cannot be guaranteed that they are absolute for a new working environment, or that they correspond to a highly accurate comparative analysis of the algorithms. However, they can certainly be seen as an indication of the agents' performance in various scenarios.

The lack of comparable experimental trajectories in Deep RL is evident in various cases, even though several quality tools exist for this reason (e.g. ALE). Even though it is true that computational costs can be a barrier for complete experimentation of all different environments included within a standardized experimental framework such as ALE, it is of utmost significance that a state-of-the-art algorithm has a strong baseline to be evaluated against, as also pointed to in (Toromanoff et al., 2019). In any other case, an algorithm's theoretical guarantees are the only properties it can display as a performance indicator, although in many cases these do not present valuable performance information.

That being said, we present the performance figures produced from data given originally for each state-of-the-art Deep RL algorithm. We distinguish two basic frameworks commonly used for the evaluation procedures: ALE and MuJoCo. Consequently, model-free, model-based and modular algorithms are compared in both of these frameworks, when applicable. It should be noted, however, that other frameworks are quite popular among the Deep RL community as well (e.g. DeepMind Control Suite proposed by Tassa et al., 2018, and ROBEL, proposed by Ahn et al., 2019), but the multitude of reported performance results come from ALE and MuJoCo, allowing for a wider comparison.

In several cases, the preferred performance tables were missing, so other sources for equivalent and accurate data were used when possible. For the purposes of this work, using data from sources other than the original was a legitimate choice, since they do provide an insight of an algorithm's performance. The data source for each experiment is cited next to its performance indication.

In the following subsections, we outline the comparison procedure and extracted insights from Atari and MuJoCo benchmarks respectively, for each of the main three families of Deep RL algorithms described throughout this work.

5.1 Atari Benchmarks

ALE has been widely used for benchmarking Deep RL algorithms that support discrete action spaces, allowing for a practical comparison between them. Due to the fact that reported results may have differences related to implementation details, such as the number of frames that an agent was trained for, or the number of runs that produced the average score, an effort is made in order to present these details in a visually appealing yet accurate manner.

First, we present charts for human-normalized performance (Mnih et al., 2015) in Atari games of model-free algorithms, model-based algorithms, and modular algorithms. Table 1 and Table 2 include raw scores of all algorithms for each Atari game. Human and Random scores are presented in Table 4.

Model-free algorithms comparison. The family of model-free algorithms have been widely tested in ALE environments, due to the algorithms' compatibility with the environments' discrete action spaces. In Figure 1 and Figure 2 of the Appendix, performance in each of these environments is presented in a collateral manner. Performance represents average maximum score that an agent achieves after several million steps, and, in some cases, for different seeds. The reported number of steps/frames for each algorithm is not the same, though it is referenced where applicable. Although this difference could be critical for comparison purposes in some cases, it is only indicative when an algorithm with fewer steps has achieved higher performance than an algorithm that was run for more frames. Figure 5 and Figure 4b of the Appendix represent treemap visualization of the number of wins for each algorithm in the Atari games.

Additionally, in most cases, both the *no-op* (Mnih et al., 2015) and *human starts* (Nair et al., 2015) regimes are used for the ALE experiments. Therefore, a performance figure is provided for each experimentation methodology.

Model-based algorithms comparison. The lack of performance data of state-of-the-art model-based algorithms on the ALE platform is evident, and this is a clear obstacle

in comparing agent performance within this category. From the set of aforementioned algorithms that belong to this category, to the authors' knowledge, the only available source of data is in the paper of Kaiser et al. (2019). In this work, we provide scores for SimPLe in the Atari games, but comparison tables against other algorithms are not given. For this reason, the performance of this algorithm shall be included only within the table of final results (Table 2).

Modular algorithms comparison. Although performance data is available for many of the algorithms presented in the corresponding category within this work, a disclaimer is necessary for the trustworthiness level of their direct comparison based on just maximal average scores reported in published works. Each method's unique characteristics aim at improving a different aspect of a core Deep RL algorithm, which is not shown on these performance figures (e.g. wall-clock time vs. total number of frames for a distributed module, or sample efficiency vs. total score for an exploration module). This kind of complete and highly accurate comparison would be possible if raw performance data was given for each algorithm, so that deeper insights about the capabilities of each algorithm were obtained.

Additionally, these module implementations are based on different core RL algorithms, which is another critical challenge for the purposes of this work. However, even though a superficial comparison of scores is the only means of gaining any sort of information regarding the power, efficiency, adequacy and aptitude of an algorithm currently, it is a first direction towards building a concrete and precise mindset for the strong evaluation of a Deep RL agent and its performance against state-of-the-art baselines.

The performance chart for each algorithm is given in Figure 3, while the treemap visualization representing the number of wins for each algorithm is given in Figure 5 of the Appendix. The no-op starts regime is used for most algorithms in this category, since only the authors of Ape-X (DQN) and Gorila (DQN) report performance scores for human-starts. Therefore, scores of these algorithms for this case are presented only in Table 1 and Table 2.

5.2 MuJoCo Benchmarks

The MuJoCo physics simulation platform is commonly used for training and evaluating an agent's performance on various continuous control tasks. The platform's support for continuous action and state spaces is the main reason why authors use it for testing Policy Gradient methods. As in Section 5.1, we present performance charts for comparison between model-free, model-based, and modular algorithms in various MuJoCo tasks. Raw performance scores for all algorithms altogether are given in Table 3.

Model-free algorithms comparison. Even though the number of common testbeds that the authors prefer to use in MuJoCo is not as large as in ALE, the current set of comparative data (presented in Figure 6 and Figure 8a) is enough to provide non-trivial conclusion regarding algorithm performance. However, it should be noted that available sources of data for state-of-the-art algorithm performance in these tasks are limited, and usually not practical for comparison purposes (e.g. visual graphs are provided instead of raw/numerical data, or testing occurs on a modified task (environment) that prohibits anyone from using any provided results for comparing with performance of other algorithms trained on the original task).

Model-free algorithms based on DQN are not present in these results, since their support of discrete action/state spaces poses a limitation to their evaluation in MuJoCo's continuous control tasks. Policy Gradient methods, on the other hand, tend to achieve remarkable results, as it can be seen in Figure 6 of the Appendix.

Model-based algorithms comparison. It is reasonable for model-based algorithms to be evaluated on environments with different properties, in order to challenge the capabilities of an algorithm that attempts to learn the underlying dynamics. Therefore, various works introduce unique or less common environments for their experimentation purposes, creating a comparison gap. As in ALE, performance data for MuJoCo tasks of model-based algorithms are also hard to obtain; however, a few data sources available gave us the chance to provide performance results for a variety of algorithms, which are given in Figure 7 and Figure 8b of the Appendix.

Modular algorithms comparison. Evaluation of modular Deep RL algorithms in MuJoCo tasks appear to be relatively uncommon, in contrast to the environments of Atari Games. What makes things worse is the difficulty of obtaining reported performance data, as in previous cases, proving a comparison study to be highly difficult. For these reasons, no data for the MuJoCo simulation platform for modular Deep RL algorithms is provided in this work.

6. Discussion

Having presented the technical part of the algorithms, it is necessary to provide in a conservative, yet concrete manner a few indications regarding the future directions of Deep RL, simultaneously addressing the most important roadblocks of the area. This can be thought of an opinion-based projection of Deep RL into the future, by the authors, which can allow the readers of this review to further comprehend the limitations and potential of the different kinds of the aforementioned methods.

6.1 Methodologies Comparison, Limitations and Potential

Model-free algorithms grasp the essence of Deep RL by optimizing policy directly based on gained experience, either with DQN-based methods or PG methods. PG methods have the important advantage of supporting continuous action/state spaces, in contrast to DQN methods, which are limited to discrete action/state spaces. In general, as it can be seen in Figure 6 or Figure 8a, algorithms such as SAC, TD3 and DDPG have a clear advantage over their simpler predecessors such as PPO and TRPO, constituting the best algorithm choices for a continuous control problem. As for environments with discrete action spaces, it can be observed in Figure 4 that DQN and variants have more wins in total than PG methods (in both no-op and human-start regimes). However, the PG methods achieve slightly fewer wins in the human-start regimes using only a fifth of the frames that DQN methods required. Empirically, this can be roughly interpreted that PG methods are more sample-efficient than DQN-based methods. Therefore, one could say that the most important asset of DQN-based methods is their easier comprehension of architectures and functionalities, making them suitable for early research stages or effective adaptation to different designs. PG methods, even though are overall more sample-efficient and have support for more environments, require a more advanced theoretical background in Deep RL in order for someone to be

able to implement novel architectures based on existing ones. A subsequent effect of the support for continuous spaces of PG algorithms is that they are also more suitable for environments with large action and state spaces.

A different perspective is shown by the authors of model-based algorithms, in which the algorithms use experience to learn environment dynamics first, and then using this information, reach the solution of the given task. There are more differences between model-free and model-based algorithms than similarities, with the increased complexity being on the model-based category. This is mainly due to the lack of a core architecture for these algorithms that an early-stage researcher could learn, use and expand. The different kinds of characteristics and functionalities found in such methods make it difficult for researchers to get involved with model-based algorithms in general, and instead focus on specific kinds of methodologies. With that said, model-based algorithms have a very significant advantage over model-free algorithms, which is their sample efficiency. With only a fraction of experience of a model-free agent, a model-based agent can perform equally or better on the same task. Sample efficiency is one of the main challenges in Deep RL, since an algorithm with good performance but low sample efficiency is essentially forbidden for use in environments with high error costs, and also impractical in cases where training time overshadows performance results. A quick comparison between model-free and model-based algorithms in MuJoCo tasks (Figure 6 and Figure 7) shows the superiority of model-based algorithms in terms of sample-efficiency, while maintaining descent performance which is comparable to that of model-free methods. Subsequently, the category of model-based algorithms appears to be the most promising among the rest, indicating that a focus on this area, albeit perplexing, is the most rewarding.

The unique architectures of each model-based method also poses a problem in using modular algorithms that incorporates such methods, something that can set a direction for further research. For this reason, model-free algorithms are more suitable for use with these frameworks, which, in the end, may not always perform better in terms of performance, but can acquire different abilities that are required for a specific problem (e.g. better generalization). Developing a framework that targets specific weaknesses in a set algorithms is one way to establish long-lasting practices in the future of Deep RL. Combinations of these frameworks (e.g. a core algorithm with a distributed framework along with an exploration module) are also possible and, to the authors' knowledge, have not been tested yet, which can also lead to novel research outcomes or application successes. It should be mentioned that sample inefficiency is very clear when using these frameworks (Figure 3), making them unsuitable for typical/standard use cases. Therefore, one has to be extremely thoughtful before deciding to carry on with the use of such a framework.

Apart from key technical challenges mentioned in the previous sections (e.g. convergence issues), this high-level comparison of the main methodologies highlighted some important challenges regarding Deep RL, such as the support of continuous spaces, effective optimization within large spaces, sample efficiency, effective exploration strategies, and even consistency between promising methodologies that can be used to advance and generalize algorithms in a more efficient way. Off-policy learning is also a topic of highly active research interest in (Deep) RL, since it introduces the non-trivial issue of having two different distributions during learning; one produced by the policy which generates data from the interaction with the environment (also called behavior policy), and one which corresponds

to the policy that we aim to improve (also called target policy). The difference between these two distributions can have a significant impact on model performance, since the target policy is updated using past experience gathered by the behavior policy (e.g. gradient computation in Policy Gradient methods), instead of fresh, observed data that are available in the behavior policy. This gap can lead to biased estimates, therefore correction measures emerged to tackle these issues (Chen et al., 2019).

The aforementioned challenges are key to introducing new concepts and breakthroughs in Deep RL that can further push the current standards to higher levels. However, there are also topics of interest which are not covered explicitly throughout this review due to its current structure, which prohibits their in-depth analysis. For the sake of completeness, we will briefly introduce some important notions in Deep RL.

6.2 Multi-agent Deep RL

Multi-Agent Deep RL is the extension of Multi-Agent RL (MARL) to use Deep RL methods. In a MARL setup, multiple agents exist within the same environment and either collaborate to solve the required task (cooperative agents) (OroojlooyJadid & Hajinezhad, 2019) or compete against each other in order for the best agent to reach the solution (competitive agents) (Zhang et al., 2019). The aforementioned algorithms throughout this review can be adjusted to work in such setup with slight modifications. In particular, in a collaborative environment the agents would have to learn to work together in order to optimize a joint policy and not their own independent policies, while in a cooperative environment the agents work against each other, improving their own policies in the process, until the best agent is found. Additionally, there is the case where the MARL setup is a mixture of the collaborative and competitive cases.

The foundations of MARL in an RL setting was first proposed in (Littman, 1994), formulated as a Markov games framework. Many works appeared afterwards (for example, the works by Littman, 2001 and Hu and Wellman, 2003, as well as by Lauer and Riedmiller, 2000), while advances in Deep RL allowed MARL to expand as well (Foerster et al., 2016; Gupta et al., 2017; Lowe et al., 2017; Foerster et al., 2017). However, key challenges in this field, such as information sharing between agents and the exponential increase of the joint action space between agents, still remain, but ongoing research attempts to understand and tackle these problems in interesting ways. For example, Jaques et al. (2019) propose a model closely related to the Theory of Mind (Rabinowitz et al., 2018; Premack & Woodruff, 1978), in which the agents are rewarded more when performing actions which strongly influence other agents' decisions, for the purposes of increasing their collaborative skills. However, the most prominent difficulty lying in dealing with the non-stationarity of the environment created by the participating agents due to their concurrent interactions between each other and the environment.

6.3 Adversarial Deep RL

An interesting concept in Deep RL is Adversarial RL, which emerges from the field of Multi-Agent RL. In Adversarial RL, the purpose is to train an agent to become robust to adversarial attacks, i.e. changes in environment dynamics caused by an adversary (i.e. opponent) that specifically aim to suppress the success of the primary agent (Goodfellow

et al., 2017; Uther & Veloso, 1997; Szegedy et al., 2014). Adversarial attacks have received elevated popularity recently, due to their ability to identify vulnerabilities and invalidate Deep Learning systems (Su et al., 2019), as well as for the purpose of increasing model performance and generalization abilities (Pinto et al., 2017).

Typically, adversarial problems in Machine Learning are closely related to minimax optimization problems (Myerson, 2013), where the objective is to find a Nash equilibrium in two-player zero-sum games (Von Neumann & Morgenstern, 2007). Such problems were also adapted for use under the MARL framework (Littman, 1994; Omidshafiei et al., 2017). The necessity for robustness in RL agents allowed various works to highlight the strength of Adversarial RL, since there have been various examples where state-of-the-art Deep RL systems such as DQN, TRPO, A3C performed poorly when adversaries where present in the environment (Huang et al., 2017; Behzadan & Munir, 2017). It is noteworthy that the opponents in such adversarial settings do not need to be better than the primary agent; an adversary can even have a seemingly random and uncoordinated behavior but still win against a state-of-the-art agent (Gleave et al., 2020).

7. Conclusions and Future Work

In this work, we presented a spherical overview of Deep RL and state-of-the-art algorithms that set the record in various platforms and environments, such as the Atari Learning Environment and MuJoCo simulation platform. We performed a clear and novel dissection of these models, giving insights and proposing new Deep RL implementations (e.g. core Deep RL algorithms combined with different host algorithms). Then, we proceeded with an ample analysis of functionalities and properties of included algorithms, exposing their common, as well as unique elements, subsequently giving insights regarding their suitability for both research and development purposes. Finally, we concentrated reported results from respective published works, in order to provide a general and comparative view of their performance in the aforementioned platforms.

Since the field of Deep Reinforcement Learning is not a relatively small area and continues to grow rapidly, there is space in this state-of-the-art walkthrough for several additions that are left for future work, such as the inclusion of more state-of-the-art algorithms (e.g. from other subcategories, such as evolutionary strategies, as proposed by Salimans et al., 2017), a more complete performance comparison, possibly with the authors' involvement in producing missing results, as well as more kinds of comparison tables (more benchmark environments or wall-clock times, for example).

Given the super-human performance on the aforementioned environments, and more specifically, video games, one cannot overlook the onset of an era where competitiveness in gaming shifts from being human-controlled to bot-controlled, a direct consequence of which, for example, is Go champion Lee Se-dol quitting the game (BBC News, 2019). What follows, reasonably of course, is the rise of new challenges in Deep RL and AI in general, which aim at building new foundations and protocols for AI, taking the corresponding countermeasures, and/or fusing the realm of AI with our own reality (Gaina et al., 2019).

Appendix A. ALE Performance

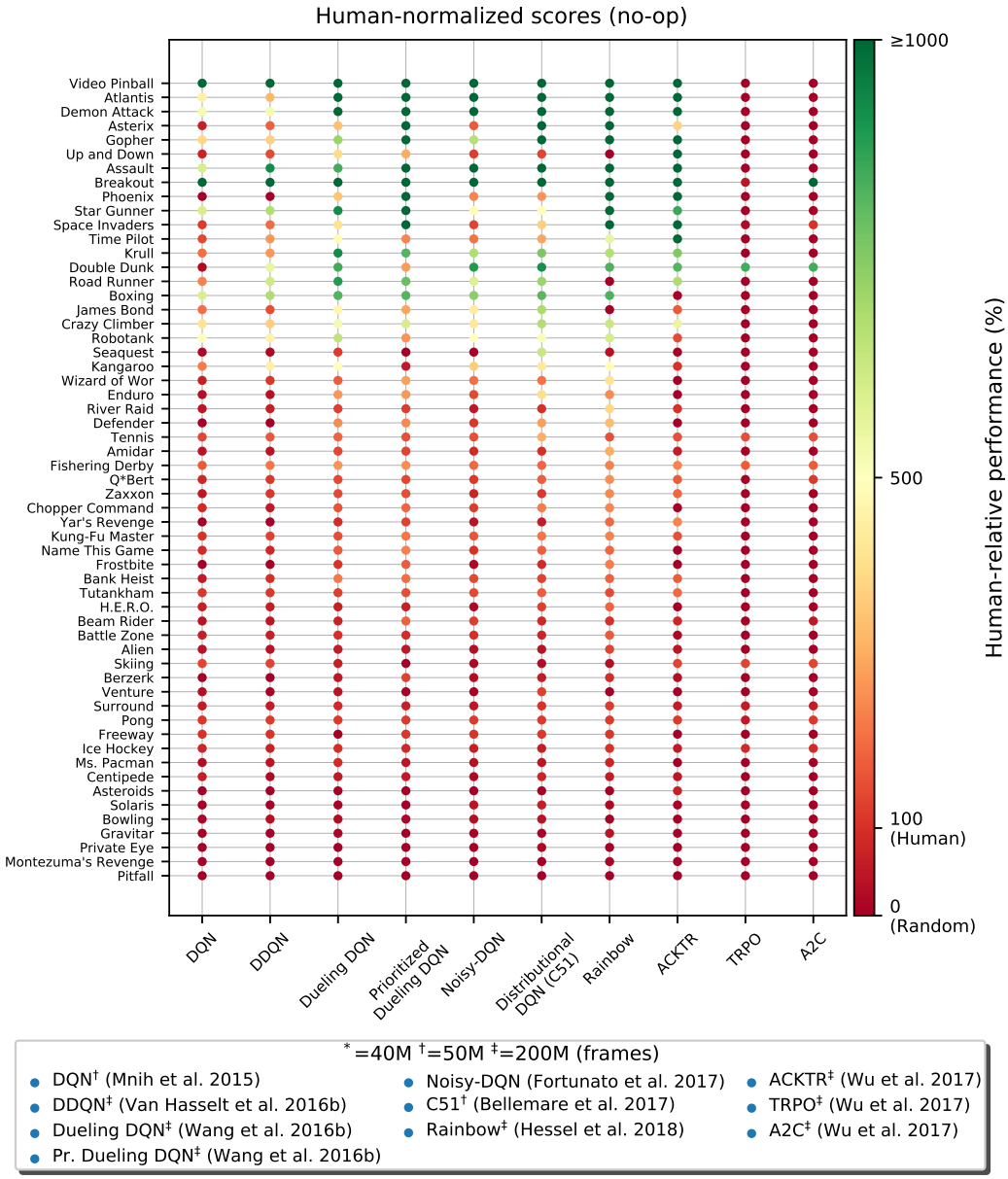


Figure 1: Heatmap visualization of performance comparison between state-of-the-art model-free algorithms in Atari games trained using the no-op regime, with respect to human performance taken from the work of Mnih et al. (2015). Scores are extracted from their original sources, where applicable. The number of frames each algorithm was run for is referenced as well, where applicable.

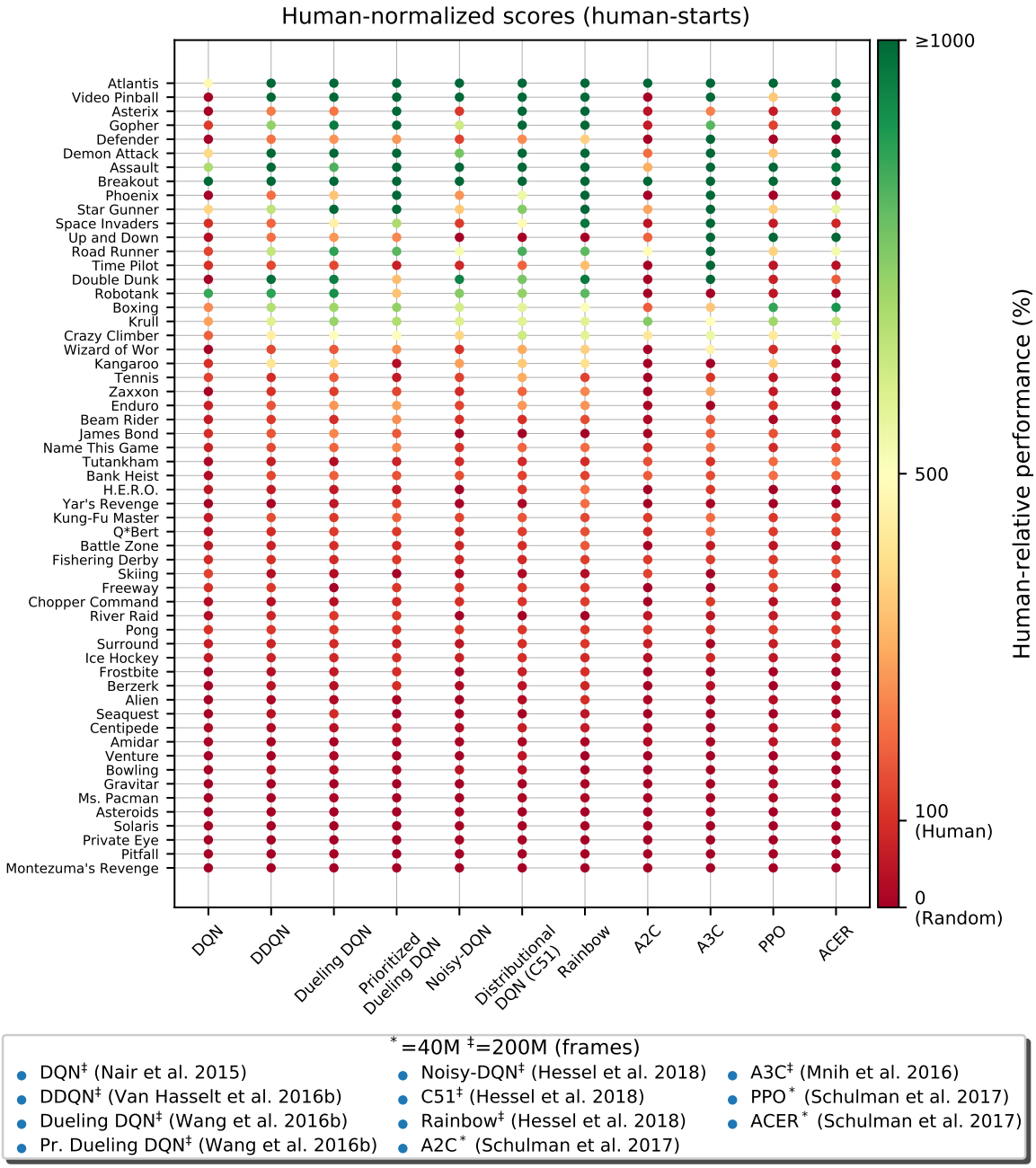


Figure 2: Heatmap visualization of performance comparison between state-of-the-art model-free algorithms in Atari games trained using the human-starts regime, with respect to human performance taken from the work of Van Hasselt et al. (2016b). Scores are extracted from their original sources, where applicable. The number of frames each algorithm was run for is referenced as well, where applicable.

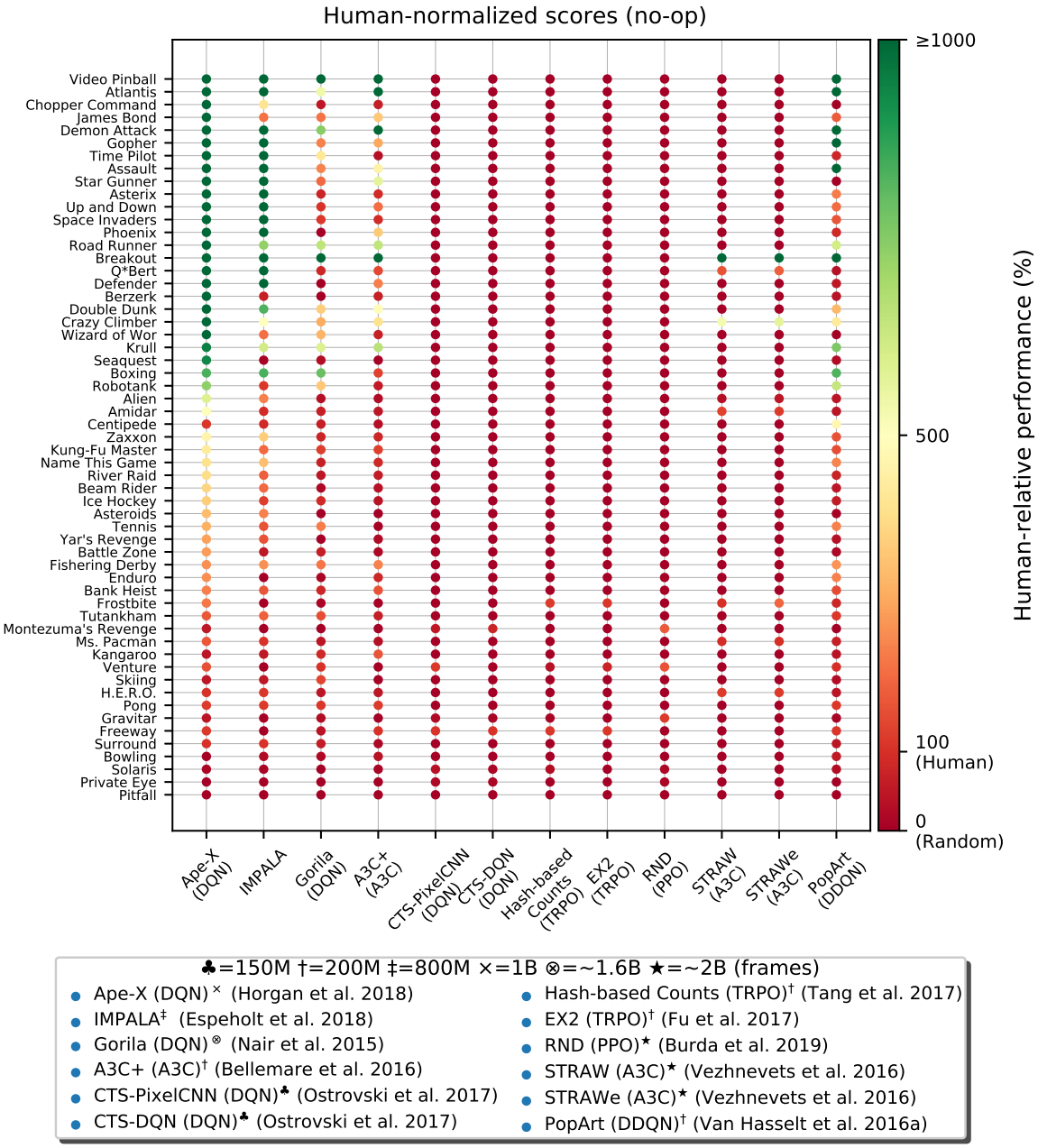
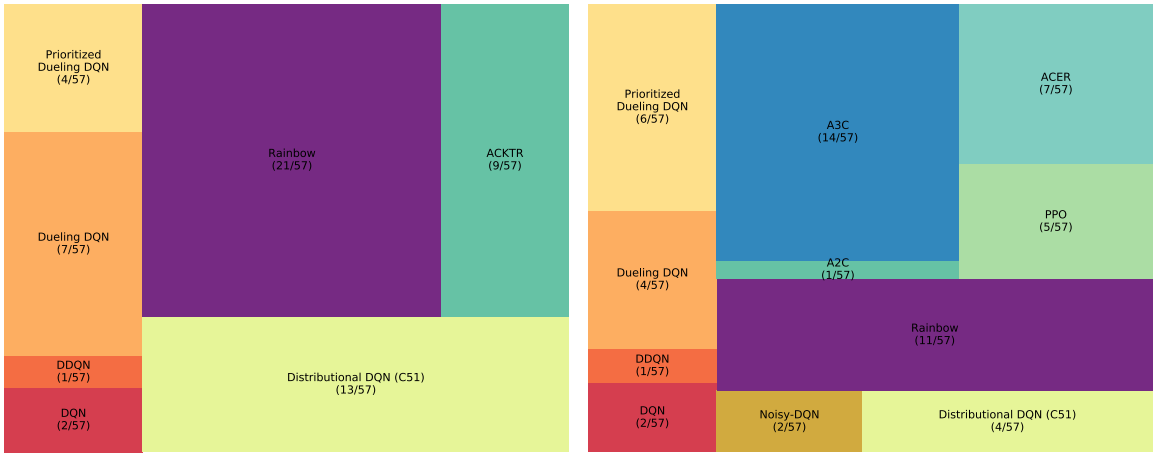


Figure 3: Heatmap visualization of performance comparison between state-of-the-art deep RL algorithms equipped with module frameworks with no-op starts, with respect to human performance taken from the work of Mnih et al. (2015). Scores are extracted from their original sources, where applicable. The number of frames each algorithm was run for is referenced as well, where applicable.



(a) (b)

Figure 4: TreeMap visualization representing the number of wins for each model-free algorithm on the ALE platform for the no-op (a) and human-start (b) regimes.

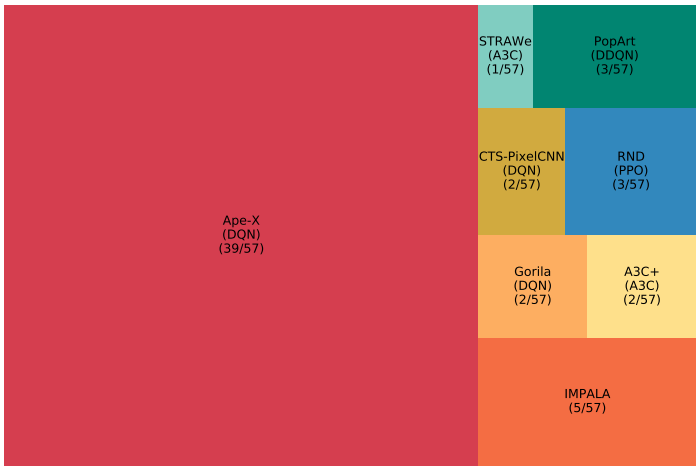


Figure 5: TreeMap visualization representing the number of wins for each modular algorithm on the ALE platform for the no-op regime.

Appendix B. MuJoCo Performance

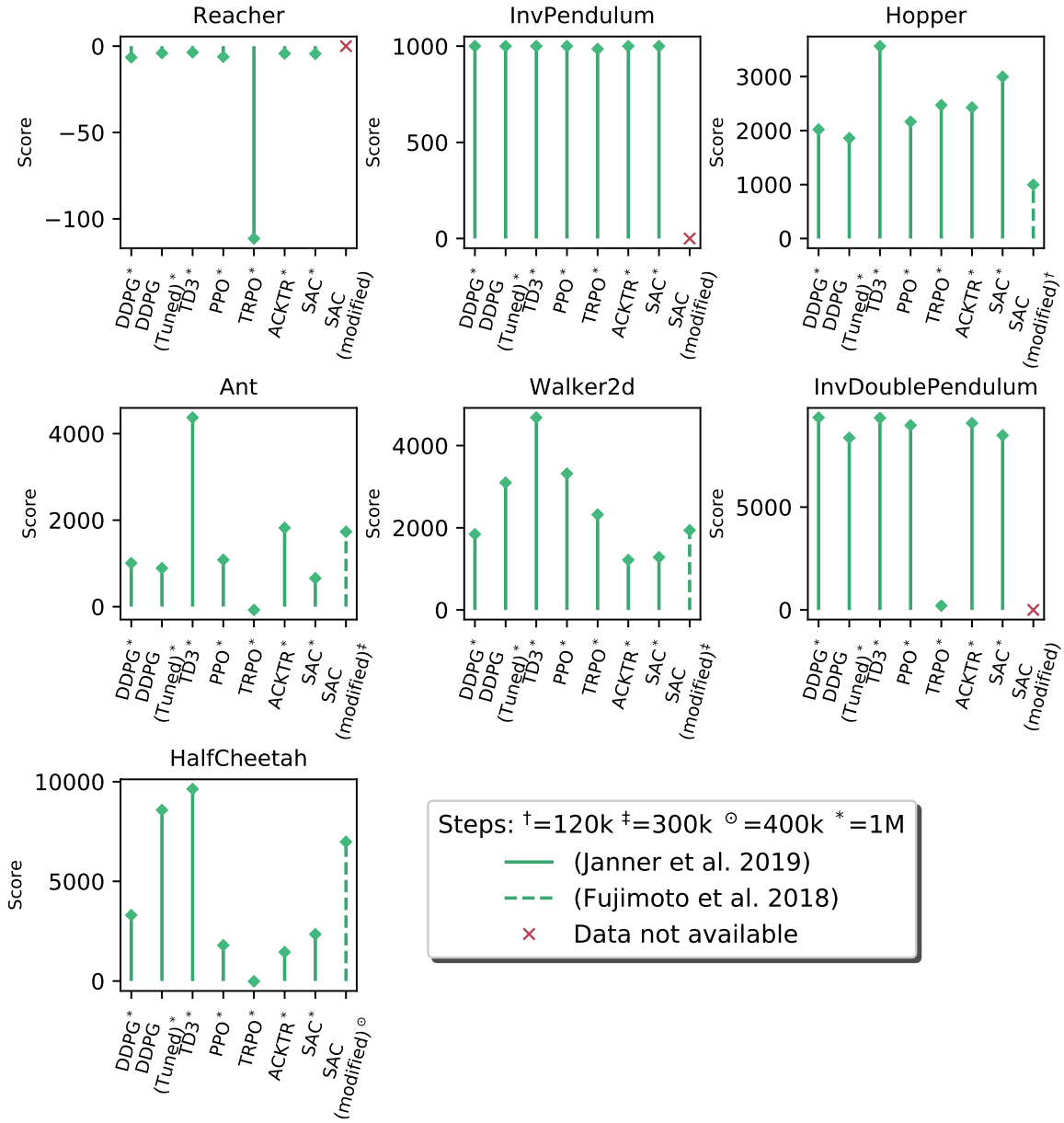


Figure 6: Performance of model-free (Policy Gradient) algorithms on various MuJoCo tasks. Data source and number of steps for each algorithm and task is referenced.

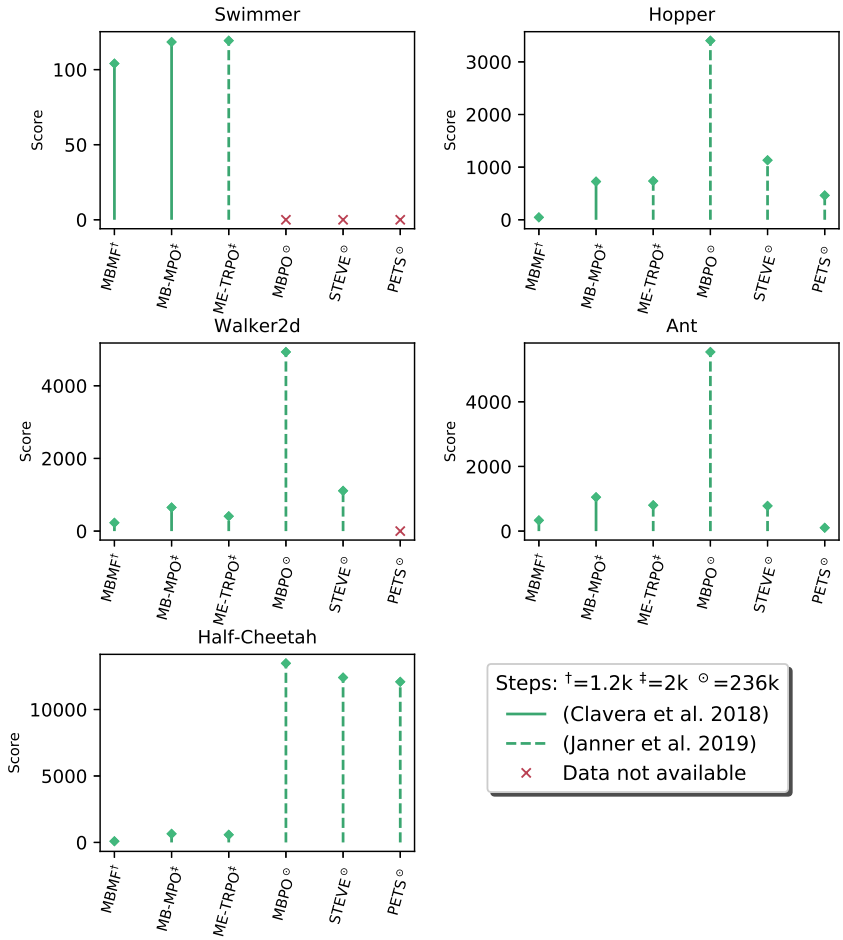
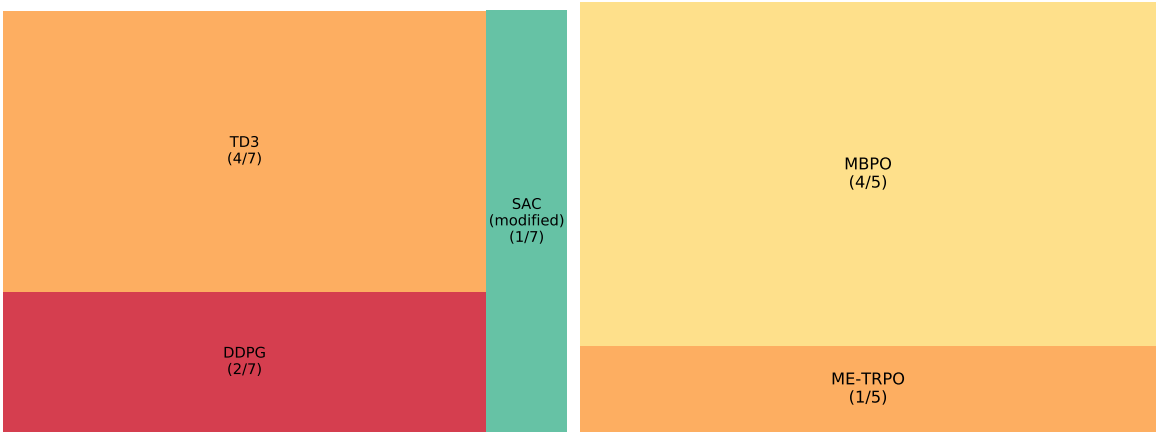


Figure 7: Performance of model-based algorithms on various MuJoCo tasks. Data source and number of steps for each algorithm and task is referenced.



(a) (b)

Figure 8: Treemap visualization representing number of wins for each algorithm of model-free (a) and model-based (b) algorithms in MuJoCo tasks, against the other algorithms.

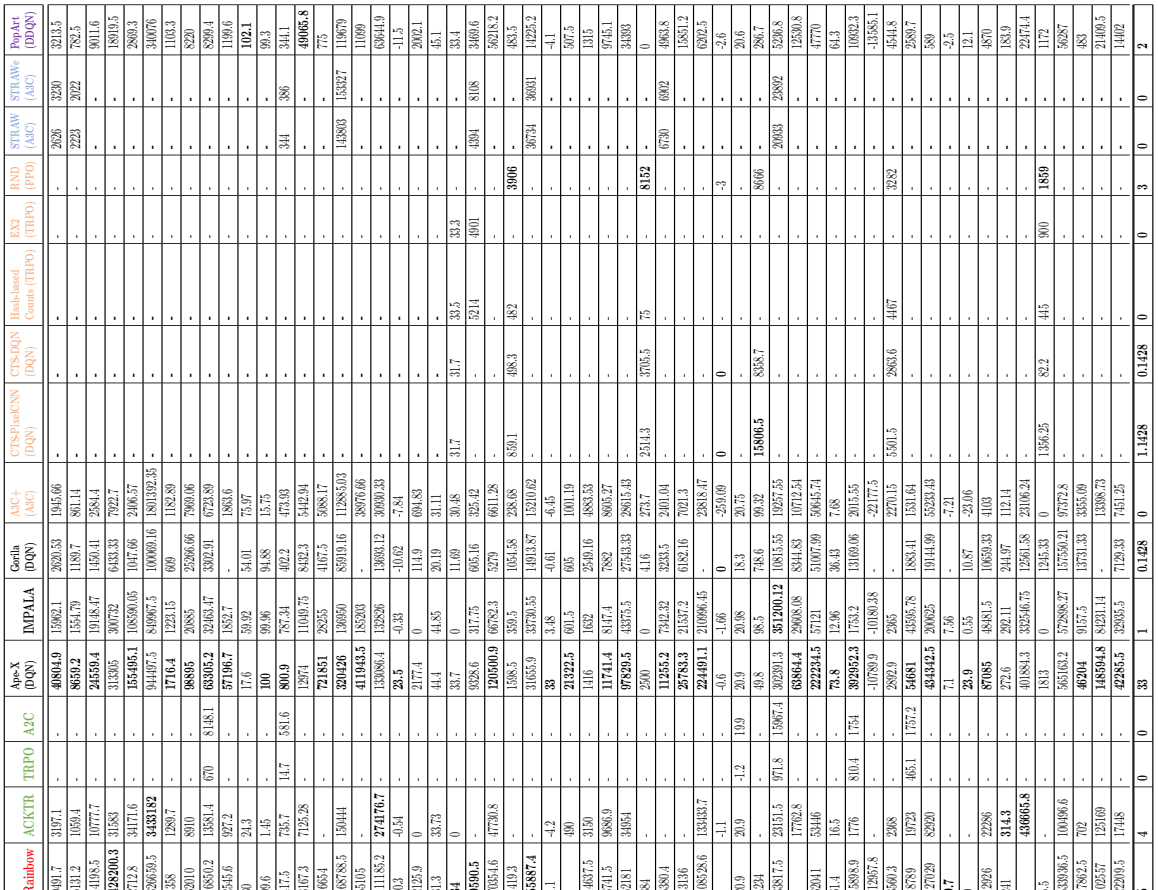
Appendix C. Tables

C.1 Raw ALE Scores (Human Starts)

Testbed	DQN	DDQN	Dueling DQN	Prioritized Dueling DQN	Noisy-DQN	Distributional DQN (C51)	Rainbow	A2C	A3C	PPO	ACER	SimPLe	Ape-X (DQN)	Gorila (DQN)
Alien	570.2	1033.4	1486.5	823.7	533.3	1997.5	6022.9	1141.7	945.3	1850.3	1655.4	616.9	**17731.5**	813.54
Amidar	133.4	169.1	172.7	238.4	148	237.7	202.8	380.8	173	674.6	827.6	74.3	**1047.3**	180.15
Assault	3332.3	6060.8	3994.8	10950.6	5124.3	5101.3	14491.7	1562.9	14497.9	4971.9	4653.8	527.2	**24404.6**	1195.85
Asterix	124.5	16837	15840	364200	8277.3	**395599.5**	280114	3176.3	17244.5	4532.5	6801.2	1128.3	283179.5	3324.7
Asteroids	697.1	1133.2	2035.4	1021.9	4078.1	2071.7	2249.4	1653.3	5093.1	2097.5	2389.3	793.6	**117303.4**	933.63
Atlantis	76108	319688	445360	423252	303666.5	289803	814684	729265.3	875822	**2311815**	1841376	20992.5	918714.5	629166.5
Bank Heist	176.3	886	1129.3	1004.6	955	835.6	826	1095.3	932.8	**1280.6**	1177.5	34.2	1200.8	399.42
Battle Zone	17560	24740	31320	30650	26985	32250	52040	3080	20760	17366.7	8983.3	4031.2	**92275**	19938
Beam Rider	8672.4	17417.2	14591.3	374122	15241.5	150024	21768.5	3031.7	24622	1590	3863.3	621.6	**72233.7**	3822.07
Berzerk	-	1011.1	910.6	2178.6	670.8	1000	1793.4	-	862.2	-	-	-	**55598.9**	-
Bowling	41.2	69.6	65.7	50.4	**79.3**	76.8	39.4	30.1	41.8	40.1	33.3	30	30.2	53.95
Boxing	25.8	73.5	77.3	79.2	66.3	62.1	54.9	17.7	37.3	94.6	**98.9**	7.8	80.9	74.2
Breakout	303.9	368.9	411.6	354.6	423.3	548.7	379.5	303	**766.8**	274.8	456.4	16.4	756.5	313.03
Centipede	3773.1	3853.5	4881	5570.2	4214.4	7476.9	7160.9	3496.5	1997	4386.4	**8904.8**	-	5711.6	6296.87
Chopper Command	3046	3496	3784	8058	8778.5	9600.5	10916	1171.7	10150	3516.3	5287.7	979.4	**576601.5**	3191.75
Crazy Climber	50992	113782	124566	127853	98576.5	154416.5	143962	107770	128518	110202	132461	62583.6	**263953.5**	65451
Defender	-	27510	33996	34415	18037.5	32246	47671.3	-	233021.5	-	-	-	**399865.3**	-
Demon Attack	12835.2	60803.4	56322.8	73371.3	25207.8	109856.6	109670.7	6639.1	115201.9	11378.4	38808.3	208.1	**133002.1**	14880.13
Double Dunk	-21.6	-0.3	-0.8	-10.7	-1	-3.7	-0.6	-16.2	0.1	-14.9	-13.2	-	**22.3**	-11.35
Enduro	475.6	1216.6	2077.4	**2223.9**	1024.5	2133.4	2061.1	0	-82.5	758.3	0	-	2042.4	71.04
Fishering Derby	-2.3	3.2	-4.1	17	-3.7	-4.9	22.6	20.6	22.6	17.8	**34.7**	-90.7	22.4	4.64
Freeway	25.8	28.8	0.2	28.8	27.1	28.8	29.1	0	0.1	**32.5**	0	16.7	29	10.16
Frostbite	157.4	1448.1	2332.4	4038.4	418.8	2813.9	4141.1	261.8	197.6	314.2	285.6	236.9	**6511.5**	426.6
Gopher	2731.8	15253	20051.4	105148.4	13131	27778.3	72595.7	15009	17106.8	2932.9	37802.3	596.8	**121168.2**	4373.04
Gravitar	216.5	200.5	297	167	256.5	422	567.5	194	320	**737.2**	225.3	173.4	662	538.37
H.E.R.O.	12952.5	14892.5	15207.9	15459.2	2454.2	28554.2	**50496.8**	-	28889.5	-	-	2656.6	26345.3	8963.36
Ice Hockey	-3.8	-2.5	-1.3	0.5	-2.4	-0.1	-0.7	-6.4	-1.7	-4.2	-5.9	-11.6	**24**	-1.72
James Bond	348.5	573	835.5	585	-	-	-	52.3	613	560.7	261.8	100.5	**18992.3**	444
Kangaroo	2696	**11204**	10334	861	7465	9555.5	10841	45.3	125	9928.7	50	51.2	577.5	1431
Krull	3864	6796.1	8051.6	7658.6	6833.5	6757.8	6715.5	8367.4	5911.4	7942.3	7268.4	2204.8	**8592**	6363.09
Kung-Fu Master	11875	30207	24288	37484	27921	33890	28999.8	24900.3	40835	23310.3	27599.3	14862.5	**72068**	20620
Montezuma's Revenge	50	42	22	24	55	130	154	0	41	42	0.3	-	**1079**	84
Ms. Pacman	763.5	1241.3	2250.6	1007.8	1012.1	2064.1	2570.2	1626.9	850.7	2096.5	2718.5	1480	**6135.4**	1263.05
Name This Game	5439.9	8960.3	11185.1	13637.9	7186.4	11382.3	11686.5	5961.2	12093.7	6254.9	8488	2420.7	**23829.9**	9238.5
Phoenix	-	12366.5	20410.5	63597	15505	31358.3	130061.6	-	74786.7	-	-	-	**188788.5**	-
Pitfall	-	-186.7	-46.9	-243.6	-154.4	-342.8	-37.6	-55	-135.7	-32.9	**-16.9**	-	-273.3	-
Pong	16.2	19.1	18.8	18.4	18	18.9	19	19.7	10.7	**20.7**	**20.7**	12.8	18.7	16.71
Private Eye	298.2	-575.5	292.6	1277.6	**5955.4**	5717.5	1704.4	91.3	421.1	69.5	182	35	864.7	2598.55
Q*Bert	4589.8	11020.8	14175.8	14063	9176.6	15035.9	18397.6	10065.7	21307.5	14293.3	15316.6	1288.8	**380.52.1**	7089.83
River Raid	4065.3	10838.4	16569.4	16496.8	-	-	-	7653.5	6591.9	8393.6	9126.1	1957.8	**49982.8**	5310.27
Road Runner	9264	43156	58549	54630	35376.5	56086	54261	32810	73949	25076	35466	5640.6	**127111.5**	43079.8
Robotank	58.5	50.1	62	24.7	50.9	49.8	55.2	2.2	2.6	5.5	2.5	-	**68.5**	61.78
Seaquest	2793.9	14488	37361.6	1431.2	2353.1	3275.4	19176	1714.3	1326.1	1204.5	1739.5	683.3	**377179.8**	10145.85
Skiing	-	-11490.4	-11928	-18955.8	-13905.9	-13247.7	-11685.8	-	-14863.8	-	-	-	-11359.3	-
Solaris	-	810	1768.4	2808.2	2530.2	2860.7	-	-	1936.4	-	-	-	**3115.9**	-
Space Invaders	1449.7	2628.7	5993.1	8978	1697.2	6308.6	12629	744.5	23846	942.5	1213.9	-	**50699.3**	1183.29
Star Gunner	34081	58365	90804	127073	31864.5	67054.5	123853	26204	164766	32689	49817.7	-	**432958**	14919.25
Surround	-	1.9	4	-0.2	-3.1	4.5	**7**	-	-8.3	-	-	-	5.5	-
Tennis	-2.3	-7.8	4.4	-13.2	-2.1	22.6	-2.2	-22.2	-6.4	-14.8	-17.6	-	**23**	-0.69
Time Pilot	5640	6608	6601	4871	5311	7684.5	11190.5	2898	27202	4342	4175.7	-	**71543**	8267.8
Tutankham	32.4	92.2	48	108.6	123.3	124.3	126.9	206.8	144.2	254.4	**280.8**	-	127.7	118.45
Up and Down	3311.3	19868.9	24759.2	22681.3	-	-	-	17369.8	105728.7	95445	145061.4	3350.3	**347912.2**	8747.67
Venture	54	21	200	29	10.5	462	45	0	25	0	0	-	**935.5**	523.4
Video Pinball	20228.1	367823.7	110976.2	447408.6	241851.7	455052.7	506817.2	19735.9	470310.5	37389	156225.6	-	**873988.5**	112093.37
Wizard of Wor	246	6201	7054	10471	4796.5	11824.5	14631.5	859	18082	4185.3	2308.3	-	**46897**	10431
Yar's Revenge	-	6270.6	25976.5	58145.9	5487.3	8267.7	93007.9	-	5615.5	-	-	5664.3	**131701.1**	-
Zaxxon	831	8593	10164	11320	7650.5	15130	19658	16.3	23519	5008.7	29	-	**37672**	6159.4
Total wins	0	1	0	1	**2**	1	**2**	0	1	5	6	0	38	0

Table 1: Raw performance scores of model-free, model-based and modular algorithms in Atari games following the human-starts methodology. Different colors indicate different algorithm subcategories (DQN, PG, Manifest state-based, Distributed frameworks).

C.2 Raw ALE Scores (No-Op)



Testbed	DQN	DDQN	Dueling DQN	Prioritized Dueling DQN	Noisy-DQN	Distributional DQN (C51)	Rainbow	ACKTR	TRPO	A2C	Ape-X (DQN)	IMPALA	Gorila (DQN)	ARC-(ASC)	CTS-PixelCNN (DQN)	CTS-DQN (DQN)	Hash-based Counts (TRPO)	EX2 (TRPO)	RND (PPO)	STRAW (ASC)	STRAWe (ASC)	PopArt (DDQN)
Alien	3069	2907.3	4461.4	3941	2403	3166	9491.7	3197.1	-	-	**40804.9**	15962.1	2620.53	1945.66	-	-	-	-	-	2626	3220	3213.5
Amidar	739.5	702.1	2354.5	2296.8	1610	1735	5131.2	1059.4	-	-	**8659.2**	1554.79	1189.7	881.14	-	-	-	-	-	2223	2022	762.5
Assault	3359	5somethingsomething22.9	4621	11477	5510	7203	14198.5	10777.7	-	-	**24559.4**	19148.47	1450.41	2584.4	-	-	-	-	-	-	-	9011.6
Asterix	6012	15150	28188	375080	14328	406211	**428200.3**	31583	-	-	313305	300732	6433.33	7922.7	-	-	-	-	-	-	-	18919.5
Asteroids	1629	930.6	2837.7	1192.7	3455	1516	2712.8	34171.6	-	-	**155495.1**	106500.05	1047.66	2406.57	-	-	-	-	-	-	-	2869.3
Atlantis	85641	64758	382572	395762	923733	841075	826659.5	**3433182**	-	-	944497.5	849967.5	100069.16	1801392.35	-	-	-	-	-	-	-	340076
Bank Heist	429.7	728	1611.9	1568.1	1068	976	1358	1289.7	-	-	**1716.4**	1223.15	609	1182.89	-	-	-	-	-	-	-	1103.3
Battle Zone	26300	25730	37150	35520	36786	28742	62010	8910	-	-	**98895**	20885	25266.66	7969.06	-	-	-	-	-	-	-	8220
Beam Rider	6846	7654	12164	30276.5	20793	14074	16850.2	13581.4	670	8148.1	**63305.2**	32463.47	3302.91	6723.89	-	-	-	-	-	-	-	8290.4
Berzerk	-	-	1472.6	3409	905	1645	2545.6	927.2	-	-	**57196.7**	1852.7	-	1803.6	-	-	-	-	-	-	-	1199.6
Bowling	42.4	70.5	65.5	46.7	71	81.8	30	24.3	-	-	17.6	59.92	54.01	75.97	-	-	-	-	-	-	-	**102.1**
Boxing	71.83	81.7	99.4	98.9	89	97.8	99.6	1.45	-	-	100	90.96	94.88	15.75	-	-	-	-	-	-	-	99.3
Breakout	401.2	375	345.3	366	516	748	417.5	735.7	14.7	581.6	**800.9**	787.34	402.2	473.93	-	-	-	-	-	344	386	344.1
Centipede	8309	41389.4	7561.4	7857.5	4269	9646	8167.3	7125.28	-	-	12974	11049.75	8482.3	5442.94	-	-	-	-	-	-	-	**49065.8**
Chopper Command	6687	4653	11215	13185	8893	15600	16054	-	-	-	**721851**	28255	4167.5	5088.17	-	-	-	-	-	-	-	775
Crazy Climber	114103	101874	143570	162224	118305	179877	168786.5	150444	-	-	**320426**	136950	85919.16	112288.603	-	-	-	-	-	143803	153027	119679
Defender	-	-	42214	41324.5	20525	47092	55105	-	-	-	**411944.5**	185203	-	28976.66	-	-	-	-	-	-	-	11099
Demon Attack	9711	9711.9	60213.3	72878.6	36150	130955	111185.2	**274176.7**	-	-	132086.4	132826	13693.12	30930.33	-	-	-	-	-	-	-	63044.9
Double Dunk	-18.1	-6.3	0.1	-12.5	1	1.5	-0.3	-0.54	-	-	**23.5**	-0.33	-10.62	-7.84	-	-	-	-	-	-	-	-11.5
Enduro	301.8	319.5	2258.2	2306.4	1240	**3454**	2125.9	0	-	-	2177.4	0	1114.9	694.83	-	-	-	-	-	-	-	2002.1
Fishering Derby	0.8	20.3	**46.4**	41.3	11	8.9	31.3	33.73	-	-	44.4	44.85	20.19	31.11	-	-	-	-	-	-	-	45.1
Freeway	30.3	31.8	0	33	32	33.9	**34**	0	-	-	33.7	0	11.69	30.48	31.7	31.7	33.5	33.3	-	-	-	33.4
Frostbite	328.3	241.5	4672.8	7413	753	3965	**9590.5**	-	-	-	9228.6	317.75	605.16	237.42	-	-	5214	4901	-	4394	8108	3469.6
Gopher	8520	8215.4	15718.4	104368.2	14574	33641	70354.6	47730.8	-	-	123086.4	132826	13693.12	5279	6611.28	-	-	-	-	-	-	63044.9
Gravitar	306.7	170.5	588	238	447	440	1419.3	-	-	-	1598.5	359.5	1054.58	238.68	859.1	498.3	482	-	**3906**	-	-	483.5
H.E.R.O.	19950	20357	20781.2	21036.5	6246	38874	**55887.4**	-	-	-	31655.9	33730.55	14913.87	15210.62	-	-	-	-	-	36734	36931	14225.2
Ice Hockey	-1.6	-2.4	0.5	-0.4	-3	-3.5	1.1	-4.2	-	-	**33**	3.48	-0.61	-6.45	-	-	-	-	-	-	-	-4.1
James Bond	576.7	438	1312.5	812	1235	1909	-	490	-	-	**21322.5**	601.5	605	1001.19	-	-	-	-	-	-	-	507.5
Kangaroo	6740	13651	**14854**	11792	10944	12853	14847.5	3150	-	-	1416	1662	2549.16	4883.53	-	-	-	-	-	-	-	1315
Krull	3805	4396	11451.9	10374.4	8805	9735	8741.5	90686.9	-	-	**11741.4**	8147.4	7882	8605.27	-	-	-	-	-	-	-	9745.1
Kung-Fu Master	23270	29486	34294	48375	36310	48192	52181	34954	-	-	**97829.5**	43375.5	27543.33	28015.43	-	-	-	-	-	-	-	34393
Montezuma's Revenge	0	0	0	0	3	0	384	-	-	-	2500	0	4.16	273.7	2514.3	3705.5	75	-	**8152**	-	-	0
Ms. Pacman	2311	3210	6283.5	3327.3	2722	3415	5380.4	-	-	-	**11255.2**	7342.32	3233.5	2401.04	-	-	-	-	-	6730	6902	4963.8
Name This Game	7257	6997	11971.1	15572.5	8181	12542	13136	-	-	-	**25783.3**	21537.2	6182.16	7021.3	-	-	-	-	-	-	-	13851.2
Phoenix	-	-	23092.2	70324.3	16028	17490	108528.6	133433.7	-	-	**224491.1**	210996.45	-	238618.47	-	-	-	-	-	-	-	6202.5
Pitfall	-	-	0	0	0	-	-1.1	-	-	-	-0.6	-1.06	0	-259.09	0	-	-	-	-3	-	-	-2.6
Pong	18.9	**21**	**21**	20.9	**21**	20.9	20.9	20.9	-1.2	19.9	20.9	20.98	18.3	20.75	-	-	-	-	-	-	-	20.6
Private Eye	1788	6710.1	103	206	3712	15095	4234	-	-	-	49.8	98.5	748.6	99.32	**15806.5**	8358.7	-	-	-	86666	-	286.7
Q*Bert	10596	14875	19220.3	18760.3	13545	23784	33817.5	23151.5	971.8	13067.4	302891.3	**351200.12**	10815.55	19257.55	-	-	-	-	-	20033	25802	5236.8
River Raid	8316	12015.3	21162.6	20607.6	9425	17322	-	17762.8	-	-	**68864.4**	29608.08	8344.83	10712.54	-	-	-	-	-	-	-	12530.8
Road Runner	18257	48377	69524	62151	45993	55839	62041	53446	-	-	**222344.5**	57121	51007.99	50645.74	-	-	-	-	-	-	-	47770
Robotank	51.6	46.7	65.3	27.5	51	52.3	61.4	16.5	-	-	**73.8**	12.96	36.43	7.68	-	-	-	-	-	-	-	64.3
Seaquest	5286	7095	50254.2	93816	2282	266434	15898.9	1776	810.4	1754	**392952.3**	1753.2	13169.06	20155.35	-	-	-	-	-	-	-	10932.3
Skiing	-	-	**-8857.4**	-19949.9	-14763	-13001	-12957.8	-	-	-	-10789.9	-10180.38	-	-22177.5	-	-	-	-	-	-	-	-13585.1
Solaris	-	-	2250.8	133.4	6088	**8342**	3560.3	2368	-	-	2892.9	2065	-	2270.15	5501.5	2863.6	4467	-	3282	-	-	4544.8
Space Invaders	1976	31514.6	6427.3	15311.5	2186	5717	18789	19723	465.1	1757.2	**54681**	43595.78	1883.41	1531.64	-	-	-	-	-	-	-	2589.7
Star Gunner	57997	65188	89238	125117	47133	49095	127029	82920	-	-	**434342.5**	200625	19144.99	55233.43	-	-	-	-	-	-	-	589
Surround	-	-	4.4	1.2	-1	6.8	**9.7**	-	-	-	7.1	7.56	-	-7.21	-	-	-	-	-	-	-	-2.5
Tennis	2.5	1.7	5.1	0	0	23.1	0	-	-	-	**23.9**	0.55	10.87	-23.06	-	-	-	-	-	-	-	12.1
Time Pilot	5947	7964	11666	7553	7035	8229	12926	22286	-	-	**87085**	48481.5	10659.33	4103	-	-	-	-	-	-	-	4870
Tutankham	186.7	190.6	211.4	245.9	232	280	241	**314.3**	-	-	272.6	292.11	244.97	112.14	-	-	-	-	-	-	-	183.9
Up and Down	8456	16769.9	44939.6	33879.1	14255	15612	-	**436605.8**	-	-	401884.3	332546.75	12561.58	231106.24	-	-	-	-	-	-	-	22474.4
Venture	380	93	497	48	97	1520	5.5	-	-	-	1813	0	1245.33	0	1356.25	82.2	445	900	**1859**	-	-	1172
Video Pinball	42684	70009	98209.5	479197	222507	**949604**	533936.5	100496.6	-	-	565163.2	572868.27	157550.21	97372.8	-	-	-	-	-	-	-	56287
Wizard of Wor	3393	5204	7855	12352	9198	9300	17682.5	702	-	-	**46204**	9157.5	13731.33	3355.09	-	-	-	-	-	-	-	483
Yar's Revenge	-	-	49622.1	69618.1	23915	35050	102557	125169	-	-	**148594.8**	84231.14	-	13398.73	-	-	-	-	-	-	-	21409.5
Zaxxon	4977	10182	12944	13886	6920	10513	22209.5	17448	-	-	**42285.5**	32935.5	7129.93	7451.25	-	-	-	-	-	-	-	14402
Total wins	0	**0.333**	**3.4761**	**0.01428**	**0.4761**	**3.1428**	**5**	**4**	**0**	**0**	**33**	**1**	**0.1428**	**0**	**1.1428**	**0.1428**	**0**	**0**	**3**	**0**	**0**	**2**

Table 2: Raw performance scores of model-free, model-based and modular algorithms in Atari games using no-op starts. Different colors indicate different algorithm subcategories (DQN, PG, Distributed Frameworks, Exploration frameworks, Hierarchical frameworks, Generalization frameworks).

C.3 Raw MuJoCo Scores

Testbed	DDPG	DDPG (Tuned)	TD3	PPO	TRPO	ACKTR	SAC	SAC (modified)	PPO	MBMF	MB-MPO	ME-TRPO	MBPO	STEVE	PETS
HalfCheetah	3305.6	8577.29	9636.95	1795.43	-15.57	1450.46	2347.19	6983.16031	1543.196	90.13927	648.6196	578.1419	**13477.5**	12406.29	12085.09
Hopper	2020.46	1860.02	**3564.07**	2164.7	2471.3	2428.39	2996.66	996.738732	635.3967	46.11076	725.9199	736.0521	3403.304	1131.613	462.8408
Walker2d	1843.85	3098.11	4682.82	3317.69	2321.47	1216.7	1283.67	1938.83028	1162.996	229.6716	651.0438	409.4306	**4930.312**	1109.227	-
Ant	1005.3	888.77	4372.44	1083.2	-75.85	1821.94	655.35	1731.42445	-29.8715	333.1264	1049.437	801.7524	**5541.369**	779.7218	103.2698
Swimmer	-	-	-	-	-	-	-	-	-	104.0466	118.3932	**119.3033**	-	-	-
Reacher	-6.51	-4.01	**-3.6**	-6.18	-111.43	-4.26	-4.44	-	-	-	-	-	-	-	-
InvPendulum	1000	1000	1000	1000	985.4	1000	1000	-	-	-	-	-	-	-	-
InvDoublePendulum	**9355.52**	8369.95	9337.47	8977.94	205.85	9081.92	8487.15	-	-	-	-	-	-	-	-
Total wins	1.1667	0.1667	1.1667	0.1667	0	0.1667	0.1667	0	0	0	0	1	3	0	0

Table 3: Raw performance scores of model-free, model-based and modular algorithms in MuJoCo tasks. Different colors indicate different algorithm subcategories (DQN, PG, Manifest state-based).

C.4 Human/Random Score Tables for Atari Games

Testbed	No-op		Human starts	
	Random	Human	Random	Human
Alien	227.8	7127.7	128.3	6371.3
Amidar	5.8	1719	11.8	1540.4
Assault	222.4	742	166.9	628.9
Asterix	210	8503.3	164.5	7536
Asteroids	719.1	47388.7	877.1	36517.3
Atlantis	12850	29028.1	13463	26575
Bank Heist	14.2	753.1	21.7	644.5
Battle Zone	2360	37187.5	3560	33030
Beam Rider	363.9	16926.5	254.6	14961
Berzerk	123.7	2630.4	196.1	2237.5
Bowling	23.1	160.7	35.2	146.5
Boxing	0.1	12.1	-1.5	9.6
Breakout	1.7	30.5	1.6	27.9
Centipede	2090.9	12017	1925.5	10321.9
Chopper Command	811	7387.8	644	8930
Crazy Climber	10780.5	35829.4	9337	32667
Defender	2874.5	18688.9	1965.5	14296
Demon Attack	152.1	1971	208.3	3442.8
Double Dunk	-18.6	-16.4	-16	-14.4
Enduro	0	860.5	-81.8	740.2
Fishering Derby	-91.7	-38.7	-77.1	5.1
Freeway	0	29.6	0.2	25.6
Frostbite	65.2	4334.7	66.4	4202.8
Gopher	257.6	2412.5	250	2311
Gravitar	173	3351.4	245.5	3116
H.E.R.O.	1027	30826.4	1580.3	25839.4
Ice Hockey	-11.2	0.9	-9.7	0.5
James Bond	29	302.8	33.5	368.5
Kangaroo	52	3035	100	2739
Krull	1598	2665.5	1151.9	2109.1
Kung-Fu Master	258.5	22736.3	304	20786.8
Montezuma's Revenge	0	4753.3	25	4182
Ms. Pacman	307.3	6951.6	197.8	15375
Name This Game	2292.3	8049	1747.8	6796
Phoenix	761.4	7242.6	1134.4	6686.2
Pitfall	-229.4	6463.7	-348.8	5998.9
Pong	-20.7	14.6	-18	15.5
Private Eye	24.9	69571.3	662.8	64169.1
Q*Bert	163.9	13455	271.8	12085
River Raid	1338.5	17118	588.3	14382.2
Road Runner	11.5	7845	200	6878
Robotank	2.2	11.9	2.4	8.9
Seaquest	68.4	42054.7	215.5	40425.8
Skiing	-17098.1	-4336.9	-15287.4	-3686.6
Solaris	1236.3	12326.7	2047.2	11032.6
Space Invaders	148	1668.7	182.6	1464.9
Star Gunner	664	10250	697	9528
Surround	-10	6.5	-9.7	5.4
Tennis	-23.8	-8.3	-21.4	-6.7
Time Pilot	3568	5229.2	3273	5650
Tutankham	11.4	167.6	12.7	138.3
Up and Down	533.4	11693.2	707.2	9896.1
Venture	0	1187.5	18	1039
Video Pinball	16256.9	17667.9	20452	15641.1
Wizard of Wor	563.5	4756.5	804	4556
Yar's Revenge	3092.9	54576.9	1476.9	47135.2
Zaxxon	32.5	9173.3	475	8443

Table 4: Random and Human player scores in Atari games, used for the normalization of raw scores.

References

Achiam, J., Edwards, H., Amodei, D., & Abbeel, P. (2018). Variational Option Discovery Algorithms. *arXiv preprint arXiv:1807.10299*.

Ahn, M., Zhu, H., Hartikainen, K., Ponte, H., Gupta, A., Levine, S., & Kumar, V. (2019). ROBEL: RObotics BEnchmarks for Learning with Low-Cost Robots. In *Conference on Robot Learning*.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., & Zaremba, W. (2017). Hindsight Experience Replay. *Advances in Neural Information Processing Systems*, *30*, 5048–5058.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine*, *34*(6), 26–38.

Bacon, P.-L., Harb, J., & Precup, D. (2017). The Option-Critic Architecture. In *AAAI Conference on Artificial Intelligence*.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., Dhruva, T., Muldal, A., Heess, N., & Lillicrap, T. (2018). Distributed Distributional Deterministic Policy Gradients. In *International Conference on Learning Representations*.

BBC News (2019). Go Master Quits Because AI 'Cannot Be Defeated'. `https://www.bbc.com/news/technology-50573071/`.

Behzadan, V., & Munir, A. (2017). Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pp. 262–275.

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A Distributional Perspective on Reinforcement Learning. In *International Conference on Machine Learning*, pp. 449–458.

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, *47*, 253–279.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*.

Bellemare, M. G., Veness, J., & Talvitie, E. (2014). Skip Context Tree Switching. In *International Conference on Machine Learning*, pp. 1458–1466.

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015). Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179.

Bhatnagar, S., Precup, D., Silver, D., Sutton, R. S., Maei, H. R., & Szepesvári, C. (2009a). Convergent Temporal-Difference Learning with Arbitrary Smooth Function Approximation. In *Advances in Neural Information Processing Systems*, pp. 1204–1212.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., & Lee, M. (2009b). Natural Actor-Critic Algorithms. *Automatica*, *45*(11), 2471–2482.

Bloom, B. H. (1970). Space/Time Trade-Offs in Hash Coding with Allowable Errors. *Communications of the ACM*, *13*(7), 422–426.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight Uncertainty in Neural Networks. In *International Conference on Machine Learning*, pp. 1613–1622.

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., & Lee, H. (2018). Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion. In *Advances in Neural Information Processing Systems*, pp. 8224–8234.

Burda, Y., Edwards, H., Storkey, A., & Klimov, O. (2019). Exploration by Random Network Distillation. In *International Conference on Learning Representations*.

Cabi, S., Colmenarejo, S. G., Hoffman, M. W., Denil, M., Wang, Z., & Freitas, N. (2017). The Intentional Unintentional Agent: Learning to Solve Many Continuous Control Tasks Simultaneously. In *Conference on Robot Learning*, pp. 207–216.

Chalapathy, R., & Chawla, S. (2019). Deep Learning for Anomaly Detection: A Survey. *arXiv preprint arXiv:1901.03407*.

Charikar, M. S. (2002). Similarity Estimation Techniques from Rounding Algorithms. In *ACM symposium on Theory of computing*, pp. 380–388.

Chen, M., Beutel, A., Covington, P., Jain, S., Belletti, F., & Chi, E. H. (2019). Top-K Off-Policy Correction for a REINFORCE Recommender System. In *ACM International Conference on Web Search and Data Mining*, pp. 456–464.

Chiappa, S., Racaniere, S., Wierstra, D., & Mohamed, S. (2017). Recurrent Environment Simulators. In *International Conference on Learning Representations*.

Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018). Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765.

Clavera, I., Rothfuss, J., Schulman, J., Fujita, Y., Asfour, T., & Abbeel, P. (2018). Model-Based Reinforcement Learning via Meta-Policy Optimization. In *Conference on Robot Learning*, pp. 617–629.

Cover, T. M. (1999). *Elements of Information Theory*. John Wiley & Sons.

Da Silva, F. L., & Costa, A. H. R. (2019). A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems. *Journal of Artificial Intelligence Research*, *64*, 645–703.

Dargan, S., Kumar, M., Ayyagari, M. R., & Kumar, G. (2019). A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Archives of Computational Methods in Engineering*, 1–22.

Dayan, P., & Hinton, G. E. (1992). Feudal Reinforcement Learning. In *Neural Information Processing Systems*, pp. 271–278.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., et al. (2012). Large Scale Distributed Deep Networks. In *Advances in Neural Information Processing Systems*, pp. 1223–1231.

Deisenroth, M. P., Neumann, G., & Peters, J. (2013). A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics*, *2*(1–2), 1–142.

Deisenroth, M. P., & Rasmussen, C. E. (2011). PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *International Conference on Machine Learning*, pp. 465–472.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., & Abbeel, P. (2016). Benchmarking Deep Reinforcement Learning for Continuous Control. In *International Conference on Machine Learning*, pp. 1329–1338.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. (2018). IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *International Conference on Machine Learning*, pp. 1407–1416.

Eysenbach, B., Gupta, A., Ibarz, J., & Levine, S. (2019). Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Learning Representations*.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., & Levine, S. (2018). Model-Based Value Estimation for Efficient Model-Free Reinforcement Learning. *arXiv preprint arXiv:1803.00101*.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., & Wierstra, D. (2017). Pathnet: Evolution Channels Gradient Descent in Super Neural Networks. *arXiv preprint arXiv:1701.08734*.

Fernando, C., Vasas, V., Szathmáry, E., & Husbands, P. (2011). Evolvable Neuronal Paths: A Novel Basis for Information and Search in the Brain. *PloS one*, *6*(8), e23534.

Finn, C., Abbeel, P., & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning*, pp. 1126–1135.

Foerster, J., Assael, I. A., De Freitas, N., & Whiteson, S. (2016). Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pp. 2137–2145.

Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., & Whiteson, S. (2017). Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*, pp. 1146–1155.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., et al. (2018). Noisy Networks For Exploration. In *International Conference on Learning Representations*.

Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., Pineau, J., et al. (2018). An Introduction to Deep Reinforcement Learning. *Foundations and Trends® in Machine Learning*, *11*(3-4), 219–354.

Frans, K., Ho, J., Chen, X., Abbeel, P., & Schulman, J. (2018). Meta Learning Shared Hierarchies. In *International Conference on Learning Representations*.

French, R. M. (1994). Catastrophic Interference in Connectionist Networks: Can It Be Predicted, Can It Be Prevented?. In *Advances in Neural Information Processing Systems*, pp. 1176–1177.

Fu, J., Co-Reyes, J., & Levine, S. (2017). EX2: Exploration with Exemplar Models for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pp. 2577–2587.

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*, pp. 1587–1596.

Gaina, R. D., Lucas, S. M., & Perez-Liebana, D. (2019). Project Thyia: A Forever Gameplayer. In *2019 IEEE Conference on Games (CoG)*, pp. 1–8.

Geffner, H. (2018). Model-Free, Model-Based, and General Intelligence. In *International Joint Conference on Artificial Intelligence*, pp. 10–17.

Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., & Russell, S. (2020). Adversarial Policies: Attacking Deep Reinforcement Learning. In *International Conference on Learning Representations*.

Goodfellow, I., Papernot, N., Huang, S., Duan, R., Abbeel, P., & Clark, J. (2017). Attacking Machine Learning with Adversarial Examples. `https://openai.com/blog/adversarial-example-research/`.

Graves, A. (2011). Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356.

Graves, A. (2013). Generating Sequences with Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850*.

Greensmith, E., Bartlett, P. L., & Baxter, J. (2004). Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *Journal of Machine Learning Research*, *5*, 1471–1530.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., & Wierstra, D. (2015). DRAW: A Recurrent Neural Network For Image Generation. In *International Conference on Machine Learning*, pp. 1462–1471.

Gregor, K., Rezende, D. J., & Wierstra, D. (2017). Variational Intrinsic Control. In *International Conference on Learning Representations*.

Grondman, I., Busoniu, L., Lopes, G. A., & Babuska, R. (2012). A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics*, *42*(6), 1291–1307.

Gupta, J. K., Egorov, M., & Kochenderfer, M. (2017). Cooperative Multi-Agent Control Using Deep Reinforcement Learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pp. 66–83.

Ha, D., & Eck, D. (2018). A Neural Representation of Sketch Drawings. In *International Conference on Learning Representations*.

Ha, D., & Schmidhuber, J. (2018). Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems*, pp. 2450–2462.

Haarnoja, T., Tang, H., Abbeel, P., & Levine, S. (2017). Reinforcement Learning with Deep Energy-Based Policies. In *International Conference on Machine Learning*, pp. 1352–1361.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018a). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, pp. 1861–1870.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., & Others (2018b). Soft Actor-Critic Algorithms and Applications. *arXiv preprint arXiv:1812.05905*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2019). Learning Latent Dynamics for Planning from Pixels. In *International Conference on Machine Learning*, pp. 2555–2565.

Hansen, N. (2016). The CMA Evolution Strategy: A Tutorial. *arXiv preprint arXiv:1604.00772*.

Hansen, N., & Ostermeier, A. (2001). Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary computation*, *9*(2), 159–195.

Hasselt, H. V. (2010). Double Q-Learning. In *Advances in Neural Information Processing Systems*, pp. 2613–2621.

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., & Riedmiller, M. (2018). Learning an Embedding Space for Transferable Robot Skills. In *International Conference on Learning Representations*.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep Reinforcement Learning that Matters. In *AAAI Conference on Artificial Intelligence*.

Hernandez-Garcia, J. F., & Sutton, R. S. (2019). Understanding Multi-Step Deep Reinforcement Learning: A Systematic Study of the DQN Target. *arXiv preprint arXiv:1901.07510*.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., & van Hasselt, H. (2019a). Multi-Task Deep Reinforcement Learning with PopArt. In *AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 3796–3803.

Hessel, M., van Hasselt, H., Modayil, J., & Silver, D. (2019b). On Inductive Biases in Deep Reinforcement Learning. *arXiv preprint arXiv:1907.02908*.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., & Lerchner, A. (2017). beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *International Conference on Learning Representations*.

Hinton, G. E. (2007). To Recognize Shapes, First Learn to Generate Images. *Progress in Brain Research*, *165*, 535–547.

Hochreiter, S. (2001). Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-term Dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*, 237–244.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., & Silver, D. (2018). Distributed Prioritized Experience Replay. In *International Conference on Learning Representations*.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., & Abbeel, P. (2016). VIME: Variational Information Maximizing Exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117.

Hu, J., & Wellman, M. P. (2003). Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, *4*, 1039–1069.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., & Abbeel, P. (2017). Adversarial Attacks on Neural Network Policies. In *International Conference on Learning Representations*.

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pp. 448–456.

Jaakkola, T., Jordan, M. I., & Singh, S. P. (1994). Convergence of Stochastic Iterative Dynamic Programming Algorithms. In *Advances in Neural Information Processing Systems*, pp. 703–710.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al. (2019). Human-Level Performance in 3D Multiplayer Games with Population-Based Reinforcement Learning. *Science*, *364*(6443), 859–865.

Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P., Strouse, D., Leibo, J. Z., & De Freitas, N. (2019). Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning. In *International Conference on Machine Learning*, pp. 3040–3049.

Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge university press.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. (1999). An Introduction to Variational Methods for Graphical Models. *Machine learning*, *37*(2), 183–233.

Justesen, N., Bontrager, P., Togelius, J., & Risi, S. (2019). Deep Learning for Video Game Playing. *IEEE Transactions on Games*, *12*(1), 1–20.

Kaiser, Ł., Babaeizadeh, M., Miłos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. (2019). Model-Based Reinforcement Learning for Atari. In *International Conference on Learning Representations*.

Kaiser, Ł., & Bengio, S. (2018). Discrete Autoencoders for Sequence Models. *arXiv preprint arXiv:1801.09797*.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., & Jaśkowski, W. (2016). Vizdoom: A Doom-Based AI Research Platform for Visual Reinforcement Learning. In *IEEE Conference on Computational Intelligence and Games*, pp. 1–8.

Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526.

Klimov, O. (2016). CarRacing-v0. `https://gym.openai.com/envs/CarRacing-v0/`.

Klyubin, A. S., Polani, D., & Nehaniv, C. L. (2005). Empowerment: A Universal Agent-Centric Measure of Control. In *IEEE Congress on Evolutionary Computation*, Vol. 1, pp. 128–135.

Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, pp. 1008–1014.

Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., & Tenenbaum, J. B. (2016). Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*, pp. 3675–3683.

Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86.

Kullback, S. (1959). *Information Theory and Statistics*. Wiley, New York.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., & Abbeel, P. (2018). Model-Ensemble Trust-Region Policy Optimization. In *International Conference on Learning Representations*.

Lauer, M., & Riedmiller, M. A. (2000). An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems. In *International Conference on Machine Learning*, pp. 535–542.

Legg, S., & Hutter, M. (2007). Universal Intelligence: A Definition of Machine Intelligence. *Minds and machines*, *17*(4), 391–444.

Leibfried, F., Kushman, N., & Hofmann, K. (2017). A Deep Learning Approach for Joint Video Frame and Reward Prediction in Atari Games. In *International Conference on Learning Representations*, pp. 1–17.

Levy, A., Platt, R., & Saenko, K. (2019). Learning Multi-Level Hierarchies with Hindsight. In *International Conference on Learning Representations*.

Li, Y. (2018). Deep Reinforcement Learning. *arXiv preprint arXiv:1810.06339*.

Li, Y. (2019). Reinforcement Learning Applications. *arXiv preprint arXiv:1908.06973*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous Control with Deep Reinforcement Learning. In *International Conference on Learning Representations*.

Lin, L.-J. (1992). *Reinforcement Learning for Robots Using Neural Networks*. Ph.D. thesis, Carnegie Mellon University.

Littman, M. L. (1994). Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*, pp. 157–163.

Littman, M. L. (2001). Value-Function Reinforcement Learning in Markov Games. *Cognitive Systems Research*, *2*(1), 55–66.

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Abbeel, O. P., & Mordatch, I. (2017). Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, pp. 6379–6390.

Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., & Kim, D. I. (2019). Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. *IEEE Communications Surveys & Tutorials*, *21*(4), 3133–3174.

MacKay, D. J. C. (1992). Information-Based Objective Functions for Active Data Selection. *Neural Computation*, *21*(4), 3133–3174.

Mahmood, A. R., Van Hasselt, H., & Sutton, R. S. (2014). Weighted Importance Sampling for Off-Policy Learning with Linear Function Approximation. In *Advances in Neural Information Processing Systems*, pp. 3014–3022.

Mahmud, M., Kaiser, M. S., Hussain, A., & Vassanelli, S. (2018). Applications of Deep Learning and Reinforcement Learning to Biological Data. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(6), 2063–2079.

Malisiewicz, T., Gupta, A., & Efros, A. A. (2011). Ensemble of Exemplar-SVMs for Object Detection and Beyond. In *IEEE International Conference on Computer Vision*, pp. 89–96.

Marbach, P., & Tsitsiklis, J. N. (2003). Approximate Gradient Methods in Policy-Space Optimization of Markov Reward Processes. *Discrete Event Dynamic Systems: Theory and Applications*, *13*(1-2), 111–148.

Martens, J., & Grosse, R. (2015). Optimizing Neural Networks with Kronecker-Factored Approximate Curvature. In *International Conference on Machine Learning*, pp. 2408–2417.

McGovern, A., Sutton, R. S., & Fagg, A. H. (1997). Roles of Macro-Actions in Accelerating Reinforcement Learning. In *Grace Hopper Celebration of Women in Computing*, Vol. 1317.

Merton, R. K. (1968). The Matthew Effect in Science: The Reward and Communication Systems of Science are Considered. *Science*, *159*(3810), 56–63.

Minsky, M. (1961). Steps Toward Artificial Intelligence. *Proceedings of the IRE*, *49*(1), 8–30.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. In *International Conference on Machine Learning*, pp. 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-Level Control Through Deep Reinforcement Learning. *Nature*, *518*(7540), 529–533.

Moore, A. (1991). *Efficient Memory-Based Learning for Robot Control*. Ph.D. thesis, University of Cambridge.

Munos, R., Stepleton, T., Harutyunyan, A., & Bellemare, M. (2016). Safe and Efficient Off-Policy Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pp. 1054–1062.

Myerson, R. B. (2013). *Game theory*. Harvard University Press.

Nachum, O., Lee, H., Gu, S., & Levine, S. (2018). Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pp. 3303–3313.

Nagabandi, A., Kahn, G., Fearing, R. S., & Levine, S. (2018). Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. In *IEEE International Conference on Robotics and Automation*, pp. 7559–7566.

Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., & Others (2015). Massively Parallel Methods for Deep Reinforcement Learning. *arXiv preprint arXiv:1507.04296*.

Nguyen, T. T., Nguyen, N. D., & Nahavandi, S. (2020). Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Transactions on Cybernetics*.

Nguyen, T. T., & Reddi, V. J. (2019). Deep Reinforcement Learning for Cyber Security. *arXiv preprint arXiv:1906.05799*.

Nosratabadi, S., Mosavi, A., Keivani, R., Ardabili, S., & Aram, F. (2020). State of the Art Survey of Deep Learning and Machine Learning Models for Smart Cities and Urban Sustainability. In *Engineering for Sustainable Future*, pp. 228–238.

Oh, J., Guo, X., Lee, H., Lewis, R., & Singh, S. (2015). Action-Conditional Video Prediction Using Deep Networks in Atari Games. In *Advances in Neural Information Processing Systems*, pp. 2863–2871.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., & Vian, J. (2017). Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability. In *International Conference on Machine Learning*, pp. 2681–2690.

OpenAI (2018). OpenAI Five. `https://blog.openai.com/openai-five/`.

OroojlooyJadid, A., & Hajinezhad, D. (2019). A Review of Cooperative Multi-Agent Deep Reinforcement Learning. *arXiv preprint arXiv:1908.03963*.

Osband, I., Van Roy, B., Russo, D., & Wen, Z. (2019). Deep Exploration via Randomized Value Functions. *Journal of Machine Learning Research*, *20*(124), 1–62.

Osband, I., Van Roy, B., & Wen, Z. (2016). Generalization and Exploration via Randomized Value Functions. In *International Conference on Machine Learning*, Vol. 48, pp. 2377–2386.

Ostrovski, G., Bellemare, M. G., Van Den Oord, A., & Munos, R. (2017). Count-Based Exploration with Neural Density models. In *International Conference on Machine Learning*, pp. 2721–2730.

Oudeyer, P. Y., & Kaplan, F. (2009). What Is Intrinsic Motivation? A Typology of Computational Approaches. *Frontiers in Neurorobotics*, *1*, 6.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-Driven Exploration by Self-Supervised Prediction. In *International Conference on Machine Learning*, Vol. 70, pp. 2778–2787.

Peters, J., & Schaal, S. (2006). Policy Gradient Methods for Robotics. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 2219–2225.

Pinto, L., Davidson, J., Sukthankar, R., & Gupta, A. (2017). Robust Adversarial Reinforcement Learning. In *International Conference on Machine Learning*, pp. 2817–2826.

Precup, D. (2000). *Temporal Abstraction in Reinforcement Learning*. Ph.D. thesis, University of Massachusetts Amherst.

Premack, D., & Woodruff, G. (1978). Does the Chimpanzee Have a Theory of Mind?. *Behavioral and Brain Sciences*, *1*(4), 515–526.

Rabinowitz, N. C., Perbet, F., Song, H. F., Zhang, C., Eslami, S., & Botvinick, M. (2018). Machine Theory of Mind. In *International Conference on Machine Learning*, pp. 4218–4227.

Racanière, S., Weber, T., Reichert, D. P., Buesing, L., Guez, A., Rezende, D., Badia, A. P., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis, D., Silver, D., & Wierstra, D. (2017). Imagination-Augmented Agents for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5690–5701.

Ratcliff, R. (1990). Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions. *Psychological Review*, *97*(2), 285.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*, pp. 1278–1286.

Richards, A. G. (2005). *Robust Constrained Model Predictive Control*. Ph.D. thesis, Massachusetts Institute of Technology.

Rocha, F. M., Costa, V. S., & Reis, L. P. (2020). From Reinforcement Learning Towards Artificial General Intelligence. In *World Conference on Information Systems and Technologies*, pp. 401–413.

Ross, S., Gordon, G. J., & Bagnell, J. A. (2011). A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 627–635.

Rubinstein, R. Y. (1997). Optimization of Computer Simulation Models with Rare Events. *European Journal of Operational Research*, *99*(1), 89–112.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., & Hadsell, R. (2016). Progressive Neural Networks. *arXiv preprint arXiv:1606.04671*.

Salge, C., Glackin, C., & Polani, D. (2014). Empowerment – An Introduction. In *Guided Self-Organization: Inception*, pp. 89–112. Springer Berlin Heidelberg.

Salimans, T., Ho, J., Chen, X., Sidor, S., & Sutskever, I. (2017). Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *arXiv preprint arXiv:1703.03864*.

Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015a). Universal Value Function Approximators. In *International Conference on Machine Learning*, pp. 1312–1320.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015b). Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952*.

Schmidhuber, J. (2010). Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, *2*(3), 230–247.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2019). Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *arXiv preprint arXiv:1911.08265*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust Region Policy Optimization. In *International Conference on Machine Learning*, pp. 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

Sehnke, F., Osendorfer, C., Rückstiess, T., Graves, A., Peters, J., & Schmidhuber, J. (2010). Parameter-Exploring Policy Gradients. *Neural Networks*, *23*(4), 551–559.

Shao, K., Tang, Z., Zhu, Y., Li, N., & Zhao, D. (2019). A Survey of Deep Reinforcement Learning in Video Games. *arXiv preprint arXiv:1912.10944*.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic Policy Gradient Algorithms. In *International Conference on Machine Learning*, pp. 387–395.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Van Den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the Game of Go Without Human Knowledge. *Nature*, *550*(7676), 354–359.

Silvia, P. J. (2012). Curiosity and Motivation. *The Oxford Handbook of Human Motivation*, *550*(7676), 354–359.

Skinner, G., & Walmsley, T. (2019). Artificial Intelligence and Deep Learning in Video Games - A Brief Review. In *IEEE International Conference on Computer and Communication Systems*, pp. 404–408.

Strehl, A. L., & Littman, M. L. (2008). An Analysis of Model-Based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, *74*(8), 1309–1331.

Stylianou, N., & Vlahavas, I. (2019). A Neural Entity Coreference Resolution Review. *arXiv preprint arXiv:1910.09329*.

Su, J., Vargas, D. V., & Sakurai, K. (2019). One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation*, *23*(5), 828–841.

Sutton, R. S. (1988). Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, *3*(1), 9–44.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT press.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, *112*(1-2), 181–211.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing Properties of Neural Networks. In *International Conference on Learning Representations*.

Tai, L., Zhang, J., Liu, M., Boedecker, J., & Burgard, W. (2016). A Survey of Deep Network Solutions for Learning Control in Robotics: From Reinforcement to Imitation. *arXiv preprint arXiv:1612.07139*.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., De Turck, F., & Abbeel, P. (2017). Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pp. 2753–2762.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. (2018). Deepmind Control Suite. *arXiv preprint arXiv:1801.00690*.

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A Physics Engine for Model-Based Control. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 5026–5033.

Toromanoff, M., Wirbel, E., & Moutarde, F. (2019). Is Deep Reinforcement Learning Really Superhuman on Atari?. *arXiv preprint arXiv:1908.04683*.

Tucker, G., Bhupatiraju, S., Gu, S., Turner, R., Ghahramani, Z., & Levine, S. (2018). The Mirage of Action-Dependent Baselines in Reinforcement Learning. In *International Conference on Machine Learning*, pp. 5015–5024.

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the Theory of the Brownian Motion. *Physical review*, *36*(5), 823.

Uther, W., & Veloso, M. (1997). Adversarial Reinforcement Learning. Tech. rep., Carnegie Mellon University. Unpublished.

Van Den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel Recurrent Neural Networks. In *International Conference on Machine Learning*, Vol. 48, pp. 1747–1756.

Van Hasselt, H., Guez, A., Hessel, M., Mnih, V., & Silver, D. (2016a). Learning Values Across Many Orders of Magnitude. In *Advances in Neural Information Processing Systems*, pp. 4287–4295.

Van Hasselt, H., Guez, A., & Silver, D. (2016b). Deep Reinforcement Learning with Double Q-Learning. In *AAAI Conference on Artificial Intelligence*, pp. 2094–2100.

Venkatraman, A., Capobianco, R., Pinto, L., Hebert, M., Nardi, D., & Bagnell, J. A. (2016). Improved Learning of Dynamics Models for Control. In *International Symposium on Experimental Robotics*, pp. 703–713.

Vezhnevets, A., Mnih, V., Agapiou, J., Osindero, S., Graves, A., Vinyals, O., & Kavukcuoglu, K. (2016). Strategic Attentive Writer for Learning Macro-Actions. In *Advances in Neural Information Processing Systems*, Vol. 29, pp. 3486–3494.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., & Kavukcuoglu, K. (2017). Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 3540–3549.

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., & Silver, D. (2019). AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`.

Von Neumann, J., & Morgenstern, O. (2007). *Theory of Games and Economic Behavior*. Princeton University Press.

Wainwright, M. J., & Jordan, M. I. (2008). Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*.

Wang, Y., He, H., Tan, X., & Gan, Y. (2019). Trust Region-Guided Proximal Policy Optimization. In *Advances in Neural Information Processing Systems*, pp. 626–636.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., & de Freitas, N. (2017). Sample Efficient Actor-Critic with Experience Replay. In *International Conference on Learning Representations*.

Wang, Z., Schaul, T., Hessel, M., & Lanctot, M. (2016). Dueling Network Architectures for Deep Reinforcement Learning. In *International Conference on Machine Learning*, pp. 1995–2003.

Watkins, C. (1989). *Learning from Delayed Rewards*. Ph.D. thesis, University of Cambridge.

Wen, Z. (2014). *Efficient Reinforcement Learning with Value Function Generalization*. Ph.D. thesis, Stanford University.

Whiteson, S. (2019). A Survey of Reinforcement Learning Informed by Natural Language. In *International Joint Conference on Artificial Intelligence*, pp. 6309–6317.

Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, *8*(3-4), 229–256.

Wu, C., Rajeswaran, A., Duan, Y., Kumar, V., Bayen, A. M., Kakade, S., Mordatch, I., & Abbeel, P. (2018). Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines. In *International Conference on Learning Representations*.

Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., & Ba, J. (2017a). Scalable Trust-Region Method for Deep Reinforcement Learning Using Kronecker-Factored Approximation. In *Advances in Neural Information Processing Systems*, pp. 5279–5288.

Wu, Y., Mansimov, E., Liao, S., Radford, A., & Schulman, J. (2017b). OpenAI Baselines: ACKTR & A2C. `https://openai.com/blog/baselines-acktr-a2c/`.

Yang, X., & Sun, M. (2019). A Survey on Deep Learning in Crop Planting. *IOP Conference Series: Materials Science and Engineering*, *490*(6), 062053.

Yu, C., Liu, J., & Nemati, S. (2019). Reinforcement Learning in Healthcare: A Survey. *arXiv preprint arXiv:1908.08796*.

Zhang, K., Yang, Z., & Başar, T. (2019). Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. *arXiv preprint arXiv:1911.10635*.

Ziebart, B. D., Maas, A., Bagnell, J. A., & Dey, A. K. (2008). Maximum Entropy Inverse Reinforcement Learning. In *National Conference on Artificial Intelligence*, Vol. 3, pp. 1433–1438.