Chapter 7
Challenges of Reinforcement Learning

Zihan Ding and Hao Dong

Abstract This chapter introduces the existing challenges in deep reinforcement learning research and applications, including: (1) the sample efficiency problem; (2) stability of training; (3) the catastrophic interference problem; (4) the exploration problems; (5) meta-learning and representation learning for the generality of reinforcement learning methods across tasks; (6) multi-agent reinforcement learning with other agents as part of the environment; (7) sim-to-real transfer for bridging the gaps between simulated environments and the real world; (8) large-scale reinforcement learning with parallel training frameworks to shorten the wall-clock time for training, etc. This chapter proposes the above challenges with potential solutions and research directions, as the primers of the advanced topics in the second main part of the book, including Chaps. 8–12, to provide the readers a relatively comprehensive understanding about the deficiencies of present methods, recent development, and future directions in deep reinforcement learning.

Keywords Sample efficiency · Stability · Catastrophic interference · Exploration · Meta-learning · Representation learning · Generality · Multi-agent reinforcement learning · Sim2real · Scalability

7.1 Sample Efficiency

A **sample-efficient** (or **data-efficient**) algorithm in reinforcement learning means that the algorithm can make better use of the collected samples, so that it can learn to improve the policy faster. With the same number of training samples (e.g., the time steps in reinforcement learning), a sample-efficient method can provide a

Z. Ding (✉)
Imperial College London, London, UK
e-mail: zhding@mail.ustc.edu.cn

H. Dong
Peking University, Beijing, China
e-mail: hao.dong@pku.edu.cn

H. Dong et al. (eds.), *Deep Reinforcement Learning*,
https://doi.org/10.1007/978-981-15-4095-0_7

superior performance on the learning curve or final results, compared with other "sample-inefficient" methods. Take the game Pong as an example: a normal human only needs dozens of trials to basically master the game and achieve a relatively good score. However, for present reinforcement learning algorithms (especially with model-free methods), it may need at least tens of thousands of samples to gradually learn some useful policies. This forms a crucial problem in reinforcement learning: how can we design a more efficient reinforcement learning algorithm for an agent to learn faster with fewer examples?

The importance of this problem is mostly due to the cost of real-time or real-world interactions between the agent and the environment, or even the time and energy consumption of the interactions in simulated environments at present. Most of present reinforcement learning algorithms are of such a low learning efficiency on a large-scale or continuous-space problem that a typical training process even with fast simulation still requires unbearable waiting time with current computational power. It can only be worse for real-world interactions. Potential problems of time consumption, wear and tear of equipment, safety during the exploration of reinforcement learning and risks of failure cases all make stricter requirements on the learning efficiency of reinforcement learning methods in practice.

Improving data efficiency requires either having informative prior knowledge or extracting information more efficiently from available data. Starting from these two points, there are several approaches in present literature for solving the problem of learning efficiency:

- Learning from expert demonstrations. The idea of learning from demonstrations requires an expert to provide samples with high reward values. It actually falls into an category called "imitation learning," which tries not only to mimic the expert actions but also to learn a generalized policy for handling unseen cases as well. The combination of imitation learning and reinforcement learning is actually a very promising area which has been heavily investigated in recent years for applications like the game of Go, robotic learning, etc., to alleviate the problem of low learning efficiency of reinforcement learning.

  The key of learning from expert demonstrations is to extract the underlying principles for generating good actions from the available demonstration dataset, and apply it to more general cases. More contents about learning from demonstrations are discussed in Chap. 8.
- Model-based reinforcement learning rather than model-free reinforcement learning. As introduced in the previous chapter, a model-based reinforcement learning method usually indicates that the agent not only learns a policy for predicting its action, but also learns a model of the environment for assisting the planning, thereby accelerating the learning process of the policy. The model of the environment basically contains two models: a transition model, which gives the state change after the agent makes an action, and a reward model, which determines how much reward the agent will get from the environment as a feedback of its action.

Learning an accurate model of the environment provides additional information for better evaluation of the agent's current policy, which could potentially make the learning process more efficient. However, the model-based methods have their own drawbacks. For instance, model-based methods always suffer from **"model bias"** problem in practice, i.e. the model-based methods usually inherently assume that the learned dynamic model of the environment sufficiently and accurately resembles the real one. But this may not always be true if there are only a few samples for the model to learn from, leading to an inaccurate model. It could be problematic when the policy learned together with the inaccurate or biased model is employed in the true environment.

One of the efficient model-based reinforcement learning algorithms is called PILCO (Deisenroth and Rasmussen 2011), which applies non-parametric probabilistic model Gaussian Processes (GPs) to resemble the dynamic model of the environment. It leverages the straightforward solving process of GP methods for efficient model learning, instead of using the neural network approximation. The policy evaluation and improvement are performed based on the learned probabilistic model. For a cart-double-pendulum swing up task in the real world, the PILCO method only takes about 20–30 trials to learn an effective policy for controlling, while the other methods like multi-layered perceptrons for learning a dynamic model will finally take at least hundreds of trials. However, the PILCO method has its own problems as well in that it cannot guarantee to search an optimal control as the non-convex optimization problem for learning the policy parameters, and the solving process of a GP is not scalable to high-dimensional parameter space for complicated models. Other model-based methods together with a general overview of model-based reinforcement learning are introduced in Chap. 9.

- Design more efficient learning algorithms through solving existing defects. The above two methods are trying to solve the learning efficiency problem through leveraging additional external information. If no extra information can be leveraged or the dynamic model of environments is hard to learn accurately, we should improve the efficiency of algorithms without extra information. There are usually two categories of reinforcement learning algorithms according to their updating manner: on-policy and off-policy, as described in previous chapters. The on-policy methods can evaluate the policy with less bias but larger variances, while the off-policy can leverage a large batch of randomly sampled data to achieve lower variances.

  Many advanced and efficient algorithms have been proposed in recent years. Most of them are targeted at some specific defects of conventional algorithms. For reducing the variance of policy gradients, the critic network is introduced to evaluate the action-value function in actor-critic; for scaling reinforcement learning tasks from small scale to large scale, deep neural networks are employed in DQN to improve the tabular-based Q-learning algorithm. To address the overestimation problem of the max operator in DQN updating rules, the double DQN method is proposed with an additional Q-network. For boosting exploration, a noisy DQN is proposed with parameter noise, and soft actor-critic (abbreviated as SAC, introduced in Chap. 6) is created with adaptive entropy for the probabilistic

distribution given by the policy. To extend the DQN methods from solving only the discrete tasks to continuous cases, deep deterministic policy gradient (abbreviated as DDPG, introduced in Chap. 6) algorithm is proposed. In order to stabilize the learning process of DDPG, twin-delayed DDPG (abbreviated as TD3, introduced in Chap. 6) is proposed with additional networks and delayed update schedule. To ensure a safe update in on-policy reinforcement learning policy optimization, trust-region-based algorithms like trust region policy optimization (abbreviated as TRPO, introduced in Chap. 5) are proposed. To reduce the computational time with second-order optimization in TRPO, the PPO (abbreviated as PPO, introduced in Chap. 5) algorithm is proposed with first-order approximation. For accelerating the second-order natural gradient descent methods, the algorithm actor-critic using Kronecker-factored trust region (abbreviated as ACKTR, introduced in Chap. 5) is proposed to use the Kronecker-factored method for approximating the inverse fisher information matrix in second-order optimization process. Maximum a posteriori policy optimization (MPO) (Abdolmaleki et al. 2018) and its on-policy variant V-MPO (Song et al. 2019) relate to the policy optimization in a perspective of "reinforcement learning as inference." The MPO employs probabilistic inference tools like expectation maximization (EM) for optimizing a maximum entropy reinforcement learning objective. The above algorithms are just a small proportion of the overall development in the field of reinforcement learning algorithms. We direct the readers to the literature for more algorithms for improving the efficiency and other drawbacks of reinforcement learning. At the same time, the structures of proposed reinforcement learning algorithms are becoming more and more complicated, with more flexible parameters either being learned adaptively or being manually chosen, which requires more delicate considerations in the research of reinforcement learning. Sometimes those additional hyper-parameters improve the learning performances greatly, but sometimes they could also make the learning process more sensitive, of which you should take care case by case.

- In the above cases we assume the data samples are information-rich, but the learning efficiency of reinforcement learning algorithms is low. In practice, it is usually common to see the samples' lack of useful information, especially for the sparse-reward tasks. For example, for a single binary valued success label of task completion, the intermediate samples may all have an immediate zero reward without any discrimination. The information contained in those samples are naturally scarce. In cases like this, the way to effectively explore the learning space without the reward instructions can be crucial. Techniques like hindsight experience replay (Andrychowicz et al. 2017), hierarchical learning structure (Kulkarni et al. 2016), intrinsic reward (Sukhbaatar et al. 2018), curiosity-driven exploration (Pathak et al. 2017), and other effective exploration strategies (Houthooft et al. 2016) are applied in some works. The learning efficiency in reinforcement learning is significantly affected by the exploration process due to the intrinsic properties of reinforcement learning, and effective exploration can improve the efficiency of learning from samples through gathering more informative samples. As exploration is another big challenge in

reinforcement learning, it will be discussed individually in one of the following sections.

7.2 Learning Stability

Deep reinforcement learning can be terribly unstable or stochastic. Here the term "unstable" indicates the differences of the learning performances during time for single run or for horizontal comparison across multiple runs. The unstable learning process during time shows up as the large local variances or non-monotonicity on the single learning curve, e.g. sometimes the learning performance even degrades for some reasons. And the unstable learning for different runs displays as a large difference in the performances across trials at each stage during training, resulting in large variances for horizontal comparisons.

The unstable and unpredictable properties of deep neural networks are further exacerbated in the deep reinforcement learning domain, due to shifting objective distribution, unsatisfied requirements of independent and identically distributed (i.i.d.) data, the unstable biased estimation of value function approximation, and so on. These factors lead to noise in the gradient estimators, which further causes the unstable learning performances. Different from learning a fixed training dataset in supervised learning (not considering the batch constrained reinforcement learning), reinforcement learning methods usually learn from the samples that are highly correlated. For example, the learning agent mostly takes samples explored with the policy, either by the current policy for on-policy learning or the previous policy for off-policy learning (sometimes even other policies). Samples generated during the sequential interactions between the agent and the environment can be highly correlated, which breaks the independent requirement for effective learning with neural networks. Since the value function is evaluated on the trajectories chosen by the current policy, there is a dependency relationship of value function on the policy for estimating it. Due to the policy changing over the training time, the optimization manifold of the parameterized value function changes over time as well. Considering the policy is usually stochastic for the benefits of exploration during training, the value function is even more untraceable. This ends up with the unsatisfied condition of identically distributed data for learning. The unstable learning is mostly caused by the variances in policy gradient or value function estimation. However, the biased estimation is another source of unstable performances in reinforcement learning, especially when the bias is unstable itself. For example, recall that in Chap. 2, the compatible function approximation condition needs to be satisfied so as to provide an unbiased estimation of action-value function $Q^{\pi}(s, a)$ with $Q^{w}(s, a)$. There are also several other conditions to ensure an unbiased estimation of value functions, as well as further requirements to ensure advanced reinforcement learning algorithms have accurate and correct gradients for improving the policy. However, in practice, those requirements or conditions are usually relaxed, which ends up with unstable biased estimation of value function, or large variances in the policy gradients. In

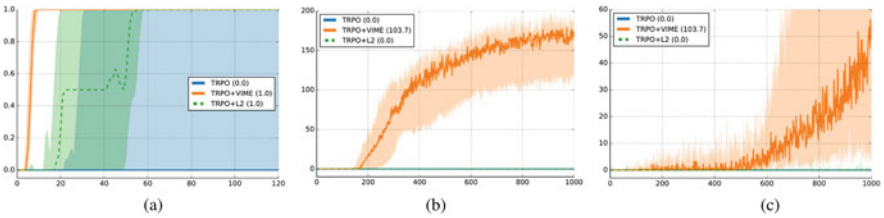

Fig. 7.1 Learning curves in experiments of VIME. Figure is adapted from Houthooft et al. (2016). (**a**) MountainCar. (**b**) CartPoleSwingup. (**c**) HalfCheetah

most cases people are talking about the bias and variance trade-off for estimation in reinforcement learning algorithms, but the unstable bias term itself can contribute in the unstable learning performances as well. There are also other factors contributing to the unstable learning performances, like randomness in the exploration strategy, randomness in the environment, random seeds for numerical calculations, etc.

Take the example of some experiments in the paper by Houthooft et al. (2016), which proposes the variational information maximizing exploration (VIME) as an exploration strategy to be applied on general reinforcement learning algorithms. Some learning performances are displayed in their comparison of algorithms, and the learning results using TRPO or TRPO+VIME for three different environments almost all show large variances in their learning curves, as shown in Fig. 7.1. For the environment MountainCar, the learning curves of TRPO algorithm could cover the whole range of reward value [0, 1], and it is almost a similar case with TRPO+VIME method for HalfCheetah environment. We need to note that TRPO is already a relatively stable reinforcement learning algorithm than other algorithms for most cases, with the second-order optimization in gradient descent and the trust-region constraint. Other algorithms like DDPG can be even more unstable during training, the noisy exploration can even deteriorate the learning performance after training for a long time (Fujimoto et al. 2018).

The randomness of the learning process with reinforcement learning makes it hard to evaluate the performance of algorithms accurately, which also addresses the importance of applying different random seeds to get averaged results.

Previous investigations (Henderson et al. 2018) about deep reinforcement learning give some conclusions about the instability and sensitivity of experiments in deep reinforcement learning:

- The policy network architecture can significantly impact results in both TRPO and DDPG.
- For hidden layers of policy network or value network, usually ReLU or leaky ReLU activations perform the best across environments and algorithms. The effects are not consistent across algorithms or environments.

- Reward rescaling can have a large effect, but results were inconsistent across environments and scaling values.
- Five random seeds (a common reporting metric) may not be enough to argue significant results, since with careful selection you can get non-overlapping confidence intervals for different random seeds even with exactly the same implementation.
- The stability of environment dynamics can severely affect the learning performance of reinforcement learning algorithms. For example, an unstable environment could diminish the effective learning performance of DDPG rapidly.

People have been working on solving the stability problem in reinforcement learning for a long time. To solve the variances in the cumulative reward function for the original REINFORCE algorithm, the value function approximation is introduced to estimate the reward value. Furthermore, the action-value function is also used for reward function approximation, which reduces the variances even if it is biased. Methods like this form a mainstream of deep reinforcement learning algorithms combining Q-learning with the policy gradient methods, as introduced in previous Chap. 6. In the original DQN (Mnih et al. 2013), the methods of using the target network with delayed update and the replay buffer help alleviate the problem of unstable learning. Usually a deep function approximator requires multiple gradient updates to converge instead of a single update, and the target network provides a stable objective during the learning process, which help with the convergence on the training data. To some extent, it satisfies the identically distributed requirement that is broken by reinforcement learning without the target networks. The replay buffer provides the DQN an off-policy learning manner, and randomly sampled data from the buffer for training is more close to the independent distributed data, which helps to stabilize the learning process as well. More details of the DQN are introduced in Chap. 4. Moreover, the TD3 algorithm (details in Chap. 6) applies the target policy smooth regularization on top of the stable techniques applied in DQN, with the smoothness assumption that similar actions should have similar value. Therefore the target value is estimated with noise on the action to reduce the variance. TD3 also employs a pair of critics instead of a single one in DDPG, the further stabilize the learning performances. On the other hand, for policy-gradient based methods, TRPO uses second-order optimization to provide more stable updates with more comprehensive information, as well as applying the constraints on updated policy to ensure conservative but steady improvements.

However, even with the above works, instability, randomness, and sensitivity to initialization and hyper-parameters make it difficult for reinforcement learning researchers to evaluate the algorithms across tasks and reproduce the results, which still forms a big challenge for the reinforcement learning community.

7.3 Catastrophic Interference

As reinforcement learning usually has a dynamics learning process instead of learning with a fixed dataset as in supervised learning, it can be regarded as a process of chasing a running goal with dataset being updated during the whole period. For example, in Chap. 2 we introduce the on-policy value function $V^{\pi}(s)$ and action-value function $Q^{\pi}(s, a)$, which are both estimated with the current policy π. But the policy is updated all the time during the learning process, which leads to a dynamic estimation of the value functions. Although applying the off-policy replay buffer helps to alleviate the problem with relatively stationary training dataset, the samples in the buffer still change along with the agent's exploration process. Therefore, a problem called **catastrophic interference** or **catastrophic forgetting** (Kirkpatrick et al. 2017) can happen during learning process especially when the policy or the value function is learnt based on the deep learning method with neural networks, and it describes the poor ability in handling this kind of incremental learning mentioned above. The new data usually makes the trained network change a lot to fit it, but forgets what it has learned in previous training process even if it is useful. This is a limitation of applying neural networks as approximators in reinforcement learning methods.

The natural and human-like learning process is actually on-policy learning, instead of the off-policy approach. Humans keep learning new things everyday in real time instead of learning from their memories all the time. However, the on-policy reinforcement learning still struggles to improve learning efficiency, and tries to prevent the catastrophic interference problem. Trust-region-based algorithms like TRPO and PPO make a constraint about the potential range of updated policy during learning, to ensure a steady but relatively slow improvement in learning performance. For on-policy learning, the data is usually collected as correlated data, which contributes to the catastrophic interference a lot. Therefore, off-policy learning methods apply an experience replay buffer for alleviating this problem, so that the old data will remain during learning to some extent. Techniques like prioritized experience replay and hindsight experience replay are proposed to leverage the data stored in the replay buffer according to their importance or goals in a sophisticated manner.

Catastrophic forgetting also happens when the learning process has multiple stages. For example, in the sim-to-real policy transfer process, the policy usually needs to be pre-trained in the simulated environment and then fine-tuned with the real-world data. However, the loss function for the two processes may be different in practice, and may not always be consistent with the overall reinforcement learning objective. Like in the work by Jeong et al. (2019a), the image observations are embedded in to latent representation as inputs of the policy, and the embedding network is fine-tuned for sim-to-real adaptation with a self-supervised loss instead of the original reinforcement learning loss in simulation training process. This kind of mismatch of the loss function in a multi-stage training process will cause catastrophic forgetting in practice, which means that the policy has chances to

forget the skills obtained in pre-training. To solve that, freezing partial layers of the network and keep updating the network with previous loss function can help during the post-training process, which tries to maintain the pre-trained network to the best during post-training process. Another similar idea is the residual policy learning mentioned in Chap. 8 Sect. 8.6, which also freezes the weights of the pre-trained network but applies an additional network alongside to learn the corrections.

7.4 Exploration

Exploration is another main challenge in reinforcement learning, which greatly affects the learning efficiency as mentioned in previous section. Rather than discussing the exploration-exploitation trade-off, which is a classical and well-known problem in reinforcement learning mentioned in Chap. 2, we focus on the challenge of exploration itself here in this section. The hardness of exploration in reinforcement learning lies in sparse rewards, large action space, and non-stationary environments for exploration, as well as the safety problems in real-world exploration, etc. Exploration means finding more information about the environments through interactions, usually counter to exploitation, which denotes exploiting known information to maximize reward. The learning process of reinforcement learning is based on trial-and-error. An optimal policy cannot be learned unless those optimal trajectories have been explored before. For example, Atari games like Montezuma's Revenge, Pitfall in OpenAI Gym are hard to solve for general reinforcement learning algorithms due to the hardness of exploration, and the game scenes of them are shown in Fig. 7.2,[1] which usually contain a complicated maze to be solved with a long sequence of operations. They are like a maze solving problem but via more complicated structures and hierarchies. Montezuma's Revenge is a very typical example with sparse rewards in the task, which makes the exploration in reinforcement learning very hard to conduct. Within one game scene, the agent in Montezuma's Revenge has to finish dozens of subsequent actions to pass one room, while there are 23 rooms of different game scenes that the agent needs to navigate itself. A wrong action at each time step could potentially make the agent fail to pass. A similar case happens in the game Pitfall. These games are usually used as a benchmark for evaluating the exploration ability of reinforcement learning methods. OpenAI[2] and Deepmind (Aytar et al. 2018) have both claimed they have solved this game Montezuma's Revenge with efficient deep reinforcement learning methods. However, the results are actually not very satisfying. In both of their solutions, the expert demonstrations are leveraged to assist exploration. For example, in Deepmind's solution they let the agent watch the YouTube videos, while OpenAI uses human demonstrations for better initialization of agent's position.

[1] Figures source: https://gym.openai.com/envs/#atari.

[2] https://openai.com/blog/learning-montezumas-revenge-from-a-single-demonstration/.

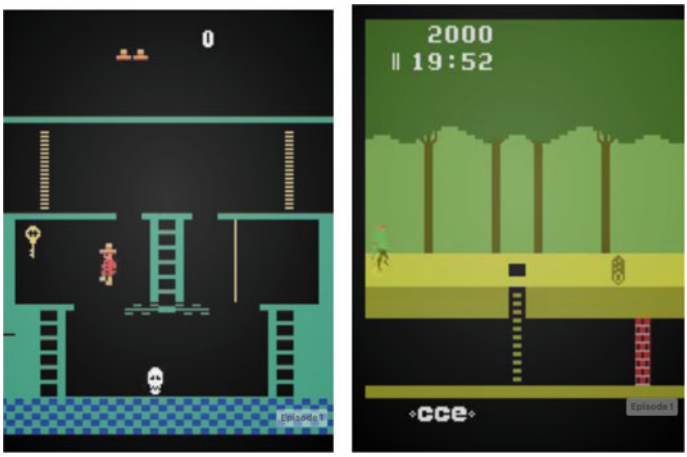

Fig. 7.2 Atari games that are hard to learn: Montezuma's revenge (left) and pitfall (right)

The bottleneck of this kind of sparse-reward task actually lies in exploration. Sparse rewards can make the value networks and policy networks optimized on hyper-surfaces that are not smooth and not convex, or even discontinuous at some stages of training. Therefore, the policy after one-step optimization may not help with exploring higher-reward regions. The agent would find it very hard to explore a high-reward trajectory during its exploration with traditional exploration strategy like random actions or ϵ-greedy policy. Even if they have sampled one near-optimal trajectory, the value-based or policy-based optimization methods may not pay enough attention to it, which could also end up with a failure or slow process of learning a good policy. The problems described above address the defects of current deep reinforcement learning methods.

Apart from the sparse rewards, large action space and non-stationary environments also raise the difficulty of exploration for reinforcement learning agents. A typical example is the StarCraft II game solved by Vinyals et al. (2019). Table 7.1[3] compares the Atari games, Go, and StarCraft in their information types, action space, moves in a game, and number of players. The large action space and length of game control sequences make it extremely hard for exploring a good policy in StarCraft. Moreover, the multi-player settings make opponents part of the game environment for the agent, which increases the hardness of exploration as well.

To solve the problem of exploration, researchers have been looking into concepts including imitation learning (as in Chap. 8), intrinsic reward/motivation, hierarchical learning (as in Chap. 10), etc. With imitation learning, the agent tries to mimic expert demonstrations from human or other sources to improve the efficiency of its learning with less difficulty in exploring near-optimal samples. Intrinsic motivation

[3]Data source: Oriol Vinyals, Deep Reinforcement Learning Workshop, NeurIPS 2019.

Table 7.1 Comparison of different games

	Atari	Go	StarCraft
Information type	Near-perfect	Perfect	Imperfect
Action space	17	361	10^{26}
Moves per game	100's	100's	1000's
Players	Single	Two	Multiple

is based on the notion that behavior is not just the result of external reward, but is also driven by internal desires, like acquiring more effective information about the unknown. For example, babies can learn about the world so fast with curiosity-driven exploration. Curiosity is one of the internal drives to improve the agent's learning towards the final goal. More internal drives are worth exploring in the research. Hierarchical learning decomposes the complicated and hard-to-explore tasks into smaller sub-tasks, which are easier to learn. For example, the feudal network (FuN) as a key method in feudal reinforcement learning applies a hierarchical structure with manager and worker to solve the Montezuma's Revenge via more effective exploration and learning (Vezhnevets et al. 2017).

In recent years, some new methods have been proposed to solve the exploration problem, one of them being Go-Explore, which is not a deep reinforcement learning solution. The main idea of Go-Explore is to first explore the game world using deterministic training without neural networks, i.e. not using deep reinforcement learning approaches, then to apply a deep neural network for imitation learning on the best trajectories, to make the policy robust to randomness of the environments. To solve the large-scale highly complicated game like StarCraft II, DeepMind's researchers (Vinyals et al. 2019) apply the population-based training framework to effectively explore the global optimal strategies, and the set of agents is called the league. Different agents are initialized around different clusters on the distribution, to ensure the diversity during exploration. The population-based training provides more thorough explorations than a single agent in the policy space.

Exploration in real-world tasks also corresponds with the safety problem. For example, when considering an autonomous driving car controlled by an agent, the failure cases with car accidents are what the agent is supposed to learn from. But an actual car cannot be used in reality to collect those failure cases for the agent to learn with a low and acceptable consumption. A real car cannot even take random actions for exploration, which could lead to disastrous results. The same problem happens in other real-world applications like robotic manipulations, robotic surgery, and so on. To solve this problem, sim-to-real transfer is developed for applying reinforcement learning in the real world, which achieves the training process in simulation and transfers the policy into reality.

7.5 Meta-Learning and Representation Learning

Apart from improving the learning efficiency on a specific task, researchers are seeking a way to improve the overall learning performance on different tasks, which relates to the **generality** and **versatility** of models. So how can we make the agent learn faster on a new task based on what it has learned from an old task? Several concepts can be introduced here, including meta-learning, representation learning, transfer learning, etc.

The problem of meta-learning can actually be traced back to 1980s–1990s (Bengio et al. 1990). Recent fast development deep learning and deep reinforcement learning bring this problem back into our sight. A lot of exciting new ideas are proposed, such as those based on model-agnostic meta-learning, and more powerful frameworks for learning across tasks are invented in recent years, which makes this area develop very fast. The original goal of meta-learning is to let the agent learn to solve different tasks or grasp different skills. However, we cannot suffer learning from scratch for each task, especially with deep learning methods for approximation. **Meta-learning**, also called **learning to learn**, is proposed to let the agent learn faster on a new task with previous experience, rather than regarding each task as an independent task. Usually a standard learner for learning a specific task is taken as an inner-loop learning process for meta-learning, while a meta-learner for learning to update the inner-loop learners is regarded as an outer-loop learning process. These two learning processes are optimized at the same time or in an iterative manner. Three main categories of meta-learning are: recurrent models, metric learning, and learning the optimizers. The combination of meta-learning and reinforcement learning gives the **meta-reinforcement learning** methods. An effective meta-reinforcement learning method like model-agnostic meta-learning (Finn et al. 2017) can solve a simple but new task with few-shot learning, or few steps for updating.

For a specific task domain, there may be some hidden correlations among different tasks. Can we enable the agent to master these underlying principles from some sampled tasks in this domain, and therefore generalize what have learned to other tasks, so as to learn them faster? Learning the underlying relationships or principles is related to a concept called **representation learning** (Bengio et al. 2013). Representation learning is originally proposed in machine learning, and is defined as learning the representations from the raw data and extract useful information or features for the classifiers or the predictors (like policies in reinforcement learning). Representation learning tries to learn some abstract and compact features to represent the raw materials, and with this kind of abstraction, the predictors or classifiers will not degrade their performances, but with a higher learning efficiency. Learning the hidden representation can be extremely useful for improving the learning efficiency of reinforcement learning, and transferring these general principles will benefit the learning process on different tasks. The representation learning is usually used for learning compact representation of complex states of reinforcement learning environments, which is called **state**

representation learning (SRL). The representation contains the invariance and distinction properties in a proper abstract space, which is distilled from variant domains. For example, in a sequence of frames of a video capturing the motion of objects, the set of the key points on the corners (or other specific points on the surfaces) of the object is an invariant and robust representation of the object motion, although the pixels in frames are always variant along with the objects' motion. Those key points are sometimes called the descriptors in computer vision terminology, within a descriptor space. Under this representation, the positions of those key points are changing during the object motion, and therefore can represent the motion of the object. Different objects will have different sets of key points, which can be used to distinguish them from each other. This area of representation learning for reinforcement learning is important when the reinforcement learning policy is transferred across domains, including different task domains, simulation-to-reality domain transfer, and so on. It is promising and still under exploration, which provides a direction for exploring how humans leverage the knowledge for planning.

7.6 Multi-Agent Reinforcement Learning

In the chapters we introduced above, there is only one agent trying to find its optimal policy in an environment, which belongs to the category of single-agent reinforcement learning. Apart from single-agent reinforcement learning, we can actually set several agents inside the same scene, to explore the policies for multi-agents at the same time in an alternating or simultaneous manner, which is called multi-agent reinforcement learning (MARL). MARL is promising and worth exploring as it provides a way to investigate the swarm intelligence, more dynamic environments for each agent, and innovations from the agents themselves, etc.

Modern learning algorithms are more so outstanding test-takers, but less so innovators. The ceiling of an agent's intelligence may be limited by the complexity of its environment. Thus, the emergence of innovation is becoming a hot topic for artificial intelligence (AI). One of the most promising paths towards such a vision is learning via social interaction with multi-agent learning. In multi-agent learning, how the agents beat the opponents or collaborate with each other is not defined by the builder of the environment. For example, the inventor of the ancient game of Go never defines what strategies are good enough to beat the opponent, but the opponent usually forms part of the dynamic environment. However, enormous and sophisticated strategies are invented while a population of human players/artificial agents evolve by improving themselves over the others, i.e. each agent is acting as an environment for the others and improving itself means proposing new problems for the others.

Combinations of traditional game theory and modern deep reinforcement learning are explored (Lanctot et al. 2017; Nowé et al. 2012) in recent years for MARL, as well as new ideas like self-play (Silver et al. 2018a; Heinrich and Silver 2016;

Shoham et al. 2003; Berner et al. 2019), prioritized fictitious self-play (Vinyals et al. 2019), population-based training (PBT) (Jaderberg et al. 2017; Vinyals et al. 2019), and independent reinforcement learning (InRL) (Tan 1993; Lanctot et al. 2017). MARL not only makes it possible to explore the distributional intelligence in a multi-agent environment, but can also help to learn the near-optimal or close-to-equilibrium agent policy in a complex large-scale environment, like in Deepmind's AlphaStar for mastering the game of StarCraft II shown in Fig. 7.3. The AlphaStar framework applies PBT, by employing a league of agents, each of which is a single colored block with index in Fig. 7.3, to ensure sufficient exploration in the policy space. The unit of policy optimization is no longer the single policy for each agent in PBT, but rather the league of agents. The overall strategy will not merely care about the improvement of a single policy, but more about the overall performances in the agent league. More contents about MARL are introduced in Chap. 11.

7.7 Sim to Real

Reinforcement learning methods can successfully solve a large variety of tasks in simulated environments, and can sometimes even beat the best human performance for specific areas as in the game of Go. However, the challenge of applying reinforcement learning methods for real-world tasks remains unsolved. Apart from playing Atari games, strategy computer games, or board games, potential applications of reinforcement learning in real world include robotics control, autonomous driving vehicles, autonomous drone control, etc. These tasks which involve real-world hardware usually have high requirements for safety and accuracy. For these cases, a single operation by mistake can even lead to disastrous results. This is a more considerable problem when the policy is learned with reinforcement learning methods, of which the exploration process makes great differences for the learning agent without even considering the sample complexity in real world. Modern machine control in industry still depends heavily on traditional control methods, instead of state-of-the-art machine learning or reinforcement learning solutions. However, it is still a wonderful dream of controlling those physical machines with a smart agent that plenty of researchers in corresponding areas are working towards.

Recent years have seen the application of deep reinforcement learning to a growing repertoire of control problems. But due to the high sample complexity of reinforcement learning algorithms and other physical limitations, many of the capabilities demonstrated in simulation have yet to be replicated in the physical world. We will demonstrate the ideas mainly with the robot learning example, which is a more and more active research direction attracting attentions from both the academia and the industry.

Guided policy search (GPS) (Levine and Koltun 2013) represents one of the few algorithms capable of training policies directly on a real robot within limited time. By leveraging trajectory optimization with learned linear dynamics models, the method is able to develop complex manipulation skills with relatively small numbers

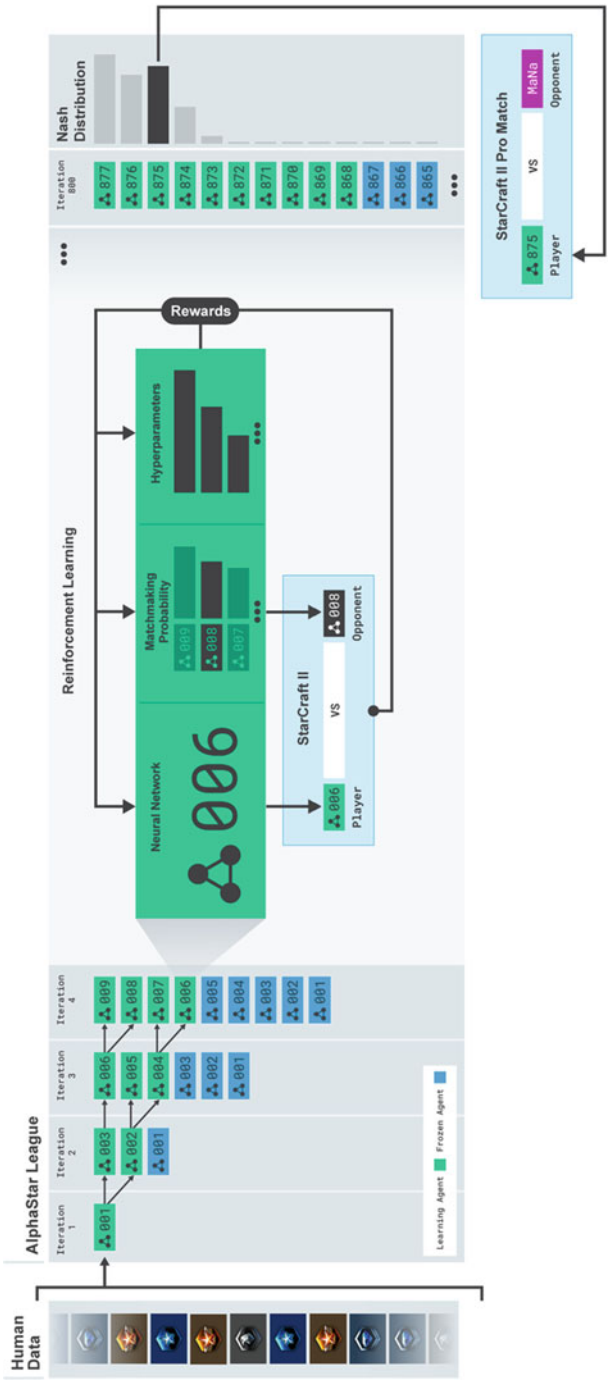

Fig. 7.3 Training scheme of AlphaStar. Each small block indicates an agent trained in the AlphaStar league

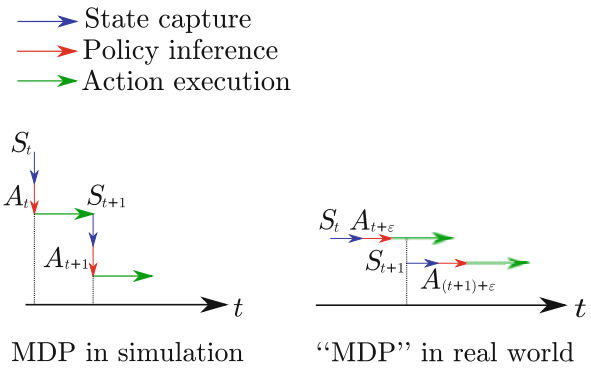

Fig. 7.4 The figure shows the difference of MDP in both simulation and in reality due to the time delays for both the state capture and the policy inference processes, which form one of the factors for reality gap

of interactions with the environment. Researchers have also explored parallelizing training across multiple robots (Levine et al. 2018). Kalashnikov et al. (2018) also propose the QT-Opt algorithm with a distributed training framework on 7 real robots at the same time, but with a cost of 800 robot hours data collection over the course of 4 months. They demonstrate the successful cases of robot learning directly deployed in real world, but the time consumption and requirement of resources are unbearable. Furthermore, successful examples of training policies directly on physical systems have so far been demonstrated only on relatively restrictive domains.

Sim-to-real transfer is an alternative approach for directly training deep reinforcement learning agents in reality, and is attracting more attention than before due to the development of simulation performances and other facts. Instead of directly training in real world, sim-to-real transfer works through a quick learning process in simulation. Recent years have seen great achievements in sim-to-real approaches for successfully deploying reinforcement learning agents in reality (Andrychowicz et al. 2018; Akkaya et al. 2019). However, the approach of sim-to-real has its own drawbacks compared with directly deploy training processes in real environments, which are mostly caused by the differences of simulation and reality environments called the **reality gap**. There are varieties of factors causing the reality gap in practice, depending on the specific systems. For example, the differences in system dynamics will cause the dynamic gap in simulation and reality (Fig. 7.4). Different approaches are also proposed to solve the problems in sim-to-real transfer, which will be discussed in later paragraphs as well.

We first try to understand the concept of the reality gap. The reality gap in real-world application can be understood to some extent with the Fig. 7.5 from the work of Jeong et al. (2019b), which displays the difference of simulated trajectories and real trajectories on robot as well as the difference of simulation and the reference. For robotic control tasks with reinforcement learning, the reference is the control signal sent by the agent or the desired behaviors on the joint angle of the robot arm. Due to the latency, inertance and other dynamic inaccuracy, both the trajectories in simulations and in reality have quite significant differences with the reference. Moreover, the trajectory in reality differs from the one in simulation as well, which

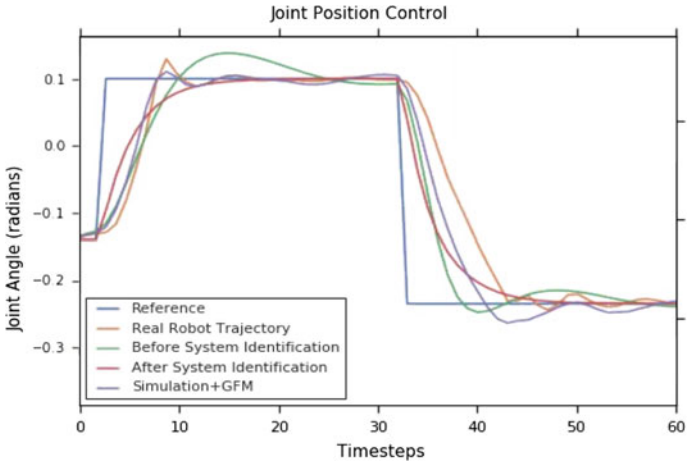

Fig. 7.5 Differences of robotic control among reference, simulation and reality, for a simple control process on the joint angle. Figure is adapted from the paper by Jeong et al. (2019b)

is the reality gap. The system identification in the graph is a method for configuring the values of dynamics parameters in the system, which can be applied by the policy or the simulator to mitigate the differences between the simulated dynamics and the real dynamics. The generalized force mode (GFM) is a newly proposed method in their paper (Jeong et al. 2019b) for calibrating the simulator with extra forces, which provides more similar trajectory in simulation as the real trajectory. However, the reality gap still exists even with the identification and calibration approaches, which will affect the transferred policy from simulation to reality.

Apart from the difference in the trajectories for simulation and reality at each time step due to different dynamic processes, there are also other sources of the reality gap. For example, the time delay of system response or system observation construction in continuous real-world control system, which may not exist in ideal simulation cases with discrete time steps. As shown in Fig. 7.4, in the MDP of simulated environments or conventional reinforcement learning settings, the state capture and policy inference process are assumed to have zero time consumption all the time, while in real-world cases, both of these two processes can take a considerable amount of time, which makes the agent always making action choices based on lagged observation from previous states during the previous action execution.

The above problem can also affect the trajectories to display different patterns for simulation and reality, as shown in Fig. 7.6. Considering an object manipulation task, even if we neglect the time consumption of the policy inference due to the fast forward process of a neural network, the position of the object in real world may need to be captured with a camera and tracked with some localization techniques, which may require some considerable time to process. This process will induce the time delay during the observation construction, and it displays the

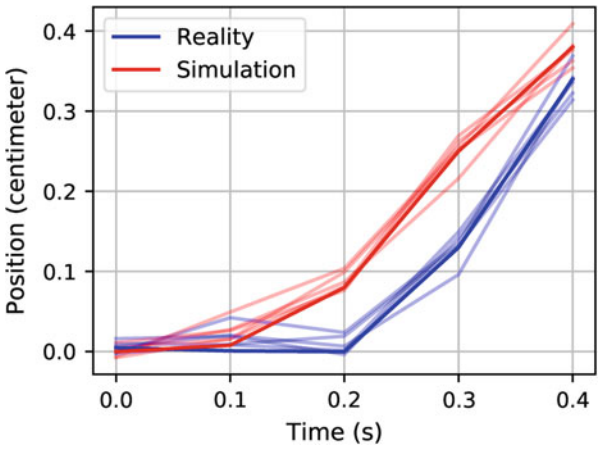

Fig. 7.6 The figure displays the time delay in the observed state (position) of the object under the same control signals. The real-world trajectory (below) is delayed compared with the simulated trajectory (above) due to the extra observation construction process in reality. Different lines show several trials and the bold ones are the means

time gap in the figure between the real-world trajectories and the simulated ones, even with the same control signals. These kind of **delayed observations** make the reinforcement learning agent in the real world only capable of receiving the previous observation O_{t-1} to make an action choice A_t for the current step instead of directly observing the current state S_t. So the policies may generally have the form $\pi(A_t|O_{t-\delta})$ according to the time delay δ in practice, which is different from the policy trained in simulation with real-time observation and therefore with a bad performance. One way of solving this is to modify the simulator to have the same time delay for the reinforcement learning agent to learn. However, this induces other problems like how to accurately represent and measure the time delay between simulation and real world, how to ensure the performance of the learned agent based on the delayed observation, etc. Recently, Ramstedt et. al. proposed real-time reinforcement learning (Ramstedt and Pal 2019) method, and Xiao et. al. proposed the method of "thinking while moving" with continuous-time MDP settings to mitigate the problem of real-time environments with delayed observations and concurrent action choices for reinforcement learning, which displays smoother trajectories for controlling in real world.

As shown above, the main problem for sim-to-real transfer in reinforcement learning perspective is: the policies trained in simulation cannot work all the time in the real world due to the reality gap, which describes the differences of simulation and reality. Due to this modeling error, policies that are successful in simulation may not be transferred well to their real-world counterparts. Generally, the methods for solving sim-to-real transfer can be divided into at least two main categories: the zero-shot methods and adaptive learning methods. The problem of transfer learning for control policies from simulation to the real world can be viewed as an instance

of **domain adaptation**, where a model trained in a source domain is transferred to a new target domain. One of the key assumptions behind these methods is that the different domains share common characteristics such that representations and behaviors learned in one will prove useful for the other. Domain adaptation requires data in the new domain to adapt the pre-trained policies in the new domain. Due to the complexity or harness for acquiring data in the new domain, e.g. collecting samples in reality, the efficiency of this kind of adaptive learning needs to be high. Methods like meta-learning (Arndt et al. 2019; Nagabandi et al. 2018) and residual policy learning (Silver et al. 2018b; Johannink et al. 2019), progressive networks (Rusu et al. 2016a,b) are applied in these scenarios. **Zero-shot** transfer is a complementary class of techniques for domain adaptation that is particularly well suited for learning in simulation. This means no further learning process on real-world data is applied during the transfer process. **Domain randomization** is one typical type of method within the category of zero-shot transfer. With domain randomization, discrepancies between the source and target domains are modeled as variability in the source domain. Instead of overfitting to the characteristics of the specific simulator settings, more general policies can be learned through domain randomization. Randomization can be applied on different characteristics according to the specific application. For example, for robotic manipulation task, the amount of friction and mass, the errors in torques and velocities will all affect the control accuracy when applied in real robot. Those parameters can therefore be randomized in simulators for training a more robust policy with reinforcement learning (Peng et al. 2018), which is called dynamics randomization. Randomization in the visual domain has been used to directly transfer vision-based policies from simulation to the real world without requiring real images during training (Sadeghi and Levine 2016; Tobin et al. 2017). Potential components for visual feature randomization include texture, lighting, objects positions, etc.

The reality gap is usually task-dependent, and it can be caused by the differences in dynamic parameters or even the definitions of dynamic process. Apart from the dynamics randomization (Peng et al. 2018) or visual feature (observation) randomization, there are some other methods for bridging the reality gap. Learning a dynamics-aware policy with system identification (Yu et al. 2017; Zhou et al. 2019) is a promising direction, which tries to learn a policy conditioned on the system characteristics like dynamics parameters or embeddings of trajectories. There are also methods trying to minimize the discrepancies between sim and real, like the GFM method mentioned previously for force calibration, etc. Sim-to-real via sim-to-sim (James et al. 2019) is another approach for crossing the reality gap using Randomized-to-Canonical Adaptation Networks (RCANs). It transforms randomized or real-world images to their equivalent non-randomized canonical versions, which are similar to ones in simulation. The progressive nets (Rusu et al. 2016a) can be applied for sim-to-real transfer (Rusu et al. 2016b), which is a general framework that enables reuse of everything from low-level visual features to high-level policies for transfer to new tasks, enabling a compositional, yet simple, approach to building complex skills.

The computational framework nowadays deploys the discrete computation process based on binary operations, so we should always admit the difference of simulation and the real world to some extent. This is because the latter is continuous in space and time (in classical physics systems at least). As long as the learning algorithms are not efficient enough to be directly applied in real world like a human's mind (or even so), it is always useful to achieve some pre-trained model in simulation. And it can be better if the model has certain level of generalization ability in real-world cases, which is the significance of the algorithms for sim-to-real transfer. In other words, the sim-to-real methods provide the methodology of learning a model always with respect to the reality gap, no matter how accurate the simulators can be.

7.8 Large-Scale Reinforcement Learning

As discussed in previous sections, reinforcement learning applications in real world suffer from several problems at present, like delayed observations, domain shifts, etc., generally within the scope of reality gap. However, there are other factors that hinder the application of reinforcement learning, either in simulated cases and in real-world cases. One of the most challenging problems is the **scalability** of reinforcement learning, although deep reinforcement learning is leveraging the general representative ability of deep neural networks. This proposes the challenge of large-scale reinforcement learning.

We can take a look at some examples first. In the applications of mastering the large-scale real-time computer games like StarCraft II and Dota 2, teams of DeepMind and OpenAI propose methods AlphaStar (Vinyals et al. 2019) and OpenAI Five (Berner et al. 2019), respectively. In AlphaStar, both deep reinforcement learning methods and supervised learning (e.g., behavioral cloning in imitation learning) are applied in a population-based training (PBT) framework, as well as advanced network structures like scatter connections, transformer, and pointer networks, which make the deep reinforcement learning methods liable to only a small fraction of the overall strategy. The steps which become more critical for finally solving the task in AlphaStar are how to efficiently learn from existing demonstration data and apply the pre-trained policy as initialization of reinforcement learning agents, and how to effectively combine different sub-optimal policies explored by different agents in the league. In OpenAI Five, a self-play framework is applied instead of the PBT framework, but it also leverages the imitation learning from human demonstration. The above facts show that, present deep reinforcement learning algorithms themselves are still not effective and efficient enough to solve a large-scale task perfectly in an end-to-end manner for most cases. Some other techniques like imitation learning (in Chap. 8), hierarchical reinforcement learning strategies (in Chap. 10), and so on are generally required for solving the large-scale problems.

Moreover, a parallel learning framework is usually employed in the large-scale problems as well. For example, in the algorithm QT-Opt (Kalashnikov et al. 2018) for solving real-world robot learning tasks, to handle the paralleled robot sampling, a replay buffer containing both on-policy and off-policy data is applied, as well as distributed training workers to learn the policy efficiently with data from the buffer. A distributed or paralleled sampling and training framework is critical for solving the large-scale problems, especially for high-dimensional state and action spaces. Espeholt et al. (2018) proposed the method called importance weighted actor-learner architecture (IMPALA) and Espeholt et al. (2019) proposed SEED (Scalable, Efficient Deep-RL) for large-scale distributed reinforcement learning. Furthermore, the distributed framework for reinforcement learning usually concerns the balance between different computational devices (e.g., CPUs and GPUs), as discussed in Chap. 18. In terms of reinforcement learning algorithms, asynchronous advantage actor-critic (A3C) (Mnih et al. 2016), distributed proximal policy optimization (DPPO) (Heess et al. 2017), recurrent replay distributed DQN (R2D2) (Kapturowski et al. 2018) are proposed in recent years for supporting better parallel sampling and training in reinforcement learning. More contents about parallel computation for reinforcement learning are introduced in Chap. 12.

7.9 Others

Apart from the above-mentioned challenges in (deep) reinforcement learning, there are also other challenges like the explainability (Madumal et al. 2019) of deep reinforcement learning, the safety problem (Berkenkamp et al. 2017; Garcıa and Fernández 2015) in applications of reinforcement learning, hardness in theoretical proofs of complexity (Lattimore et al. 2013; Koenig and Simmons 1993), efficiency (Jin et al. 2018), and convergence property (Papavassiliou and Russell 1999) for reinforcement learning algorithms, and figuring out the role of reinforcement learning methods in general artificial intelligence, etc. These contents are beyond the scope of the book, readers with interests are encouraged to explore the frontiers of these domains.

At the end of this chapter, we quote some words by Richard Sutton (2019).[4] "*One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great. The two methods that seem to scale arbitrarily in this way are search and learning.* " This is based on the observations that the previous success in computer chess and computer Go, as well as in speech recognition and computer vision, the general statistical methods (e.g., neural networks) won over the human-knowledge-based methods. So the built-in knowledge in intelligent agents may satisfy the researchers within a short term,

[4]Richard S. Sutton. "The Bitter Lesson." March 13, 2019.

but may hinder the general progress of general artificial intelligence in a long run. "*The second general point to be learned from the bitter lesson is that the actual contents of minds are tremendously, irredeemably complex; we should stop trying to find simple ways to think about the contents of minds, such as simple ways to think about space, objects, multiple agents, or symmetries. All these are part of the arbitrary, intrinsically-complex, outside world. They are not what should be built in, as their complexity is endless; instead we should build in only the meta-methods that can find and capture this arbitrary complexity.* " This emphasizes the importance of proposing meta-methods that can handle the complexity of the world naturally, rather than applying the relatively simple cognitive structures and decision-making mechanisms that are manually built by humans for special functionalities.

References

Abdolmaleki A, Springenberg JT, Tassa Y, Munos R, Heess N, Riedmiller M (2018) Maximum a posteriori policy optimisation. arXiv:180606920

Akkaya I, Andrychowicz M, Chociej M, Litwin M, McGrew B, Petron A, Paino A, Plappert M, Powell G, Ribas R, et al (2019) Solving Rubik's cube with a robot hand. arXiv:191007113

Andrychowicz M, Wolski F, Ray A, Schneider J, Fong R, Welinder P, McGrew B, Tobin J, Abbeel OP, Zaremba W (2017) Hindsight experience replay. In: Advances in neural information processing systems, pp 5048–5058

Andrychowicz M, Baker B, Chociej M, Jozefowicz R, McGrew B, Pachocki J, Petron A, Plappert M, Powell G, Ray A, et al (2018) Learning dexterous in-hand manipulation. arXiv:180800177

Arndt K, Hazara M, Ghadirzadeh A, Kyrki V (2019) Meta reinforcement learning for sim-to-real domain adaptation. arXiv:190912906

Aytar Y, Pfaff T, Budden D, Paine T, Wang Z, de Freitas N (2018) Playing hard exploration games by watching YouTube. In: Advances in neural information processing systems, pp 2930–2941

Bengio Y, Bengio S, Cloutier J (1990) Learning a synaptic learning rule. Université de Montréal, Département d'informatique et de recherche opérationnelle

Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell 35(8):1798–1828

Berkenkamp F, Turchetta M, Schoellig A, Krause A (2017) Safe model-based reinforcement learning with stability guarantees. In: Advances in neural information processing systems, pp 908–918

Berner C, Brockman G, Chan B, Cheung V, Dębiak P, Dennison C, Farhi D, Fischer Q, Hashme S, Hesse C, et al (2019) Dota 2 with large scale deep reinforcement learning. arXiv:191206680

Deisenroth M, Rasmussen CE (2011) PILCO: a model-based and data-efficient approach to policy search. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp 465–472

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. arXiv:180201561

Espeholt L, Marinier R, Stanczyk P, Wang K, Michalski M (2019) Seed RL: Scalable and efficient deep-RL with accelerated central inference. arXiv:191006591

Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th international conference on machine learning, vol 70, pp 1126–1135. JMLR.org

Fujimoto S, van Hoof H, Meger D (2018) Addressing function approximation error in actor-critic methods. arXiv:180209477

Garcıa J, Fernández F (2015) A comprehensive survey on safe reinforcement learning. J Mach Learn Res 16(1):1437–1480

Heess N, Sriram S, Lemmon J, Merel J, Wayne G, Tassa Y, Erez T, Wang Z, Eslami S, Riedmiller M, et al (2017) Emergence of locomotion behaviours in rich environments. arXiv:170702286

Heinrich J, Silver D (2016) Deep reinforcement learning from self-play in imperfect-information games. arXiv:160301121

Henderson P, Islam R, Bachman P, Pineau J, Precup D, Meger D (2018) Deep reinforcement learning that matters. In: Thirty-second AAAI conference on artificial intelligence

Houthooft R, Chen X, Duan Y, Schulman J, Turck FD, Abbeel P (2016) VIME: variational information maximizing exploration. https://1605.09674

Jaderberg M, Dalibard V, Osindero S, Czarnecki WM, Donahue J, Razavi A, Vinyals O, Green T, Dunning I, Simonyan K, et al (2017) Population based training of neural networks. arXiv:171109846

James S, Wohlhart P, Kalakrishnan M, Kalashnikov D, Irpan A, Ibarz J, Levine S, Hadsell R, Bousmalis K (2019) Sim-to-real via sim-to-sim: data-efficient robotic grasping via randomized-to-canonical adaptation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 12627–12637

Jeong R, Aytar Y, Khosid D, Zhou Y, Kay J, Lampe T, Bousmalis K, Nori F (2019a) Self-supervised sim-to-real adaptation for visual robotic manipulation. arXiv:191009470

Jeong R, Kay J, Romano F, Lampe T, Rothorl T, Abdolmaleki A, Erez T, Tassa Y, Nori F (2019b) Modelling generalized forces with reinforcement learning for sim-to-real transfer. arXiv:191009471

Jin C, Allen-Zhu Z, Bubeck S, Jordan MI (2018) Is Q-learning provably efficient? In: Advances in neural information processing systems, pp 4863–4873

Johannink T, Bahl S, Nair A, Luo J, Kumar A, Loskyll M, Ojea JA, Solowjow E, Levine S (2019) Residual reinforcement learning for robot control. In: 2019 international conference on robotics and automation (ICRA). IEEE, Piscataway, pp 6023–6029

Kalashnikov D, Irpan A, Pastor P, Ibarz J, Herzog A, Jang E, Quillen D, Holly E, Kalakrishnan M, Vanhoucke V, et al (2018) QT-opt: scalable deep reinforcement learning for vision-based robotic manipulation. arXiv:180610293

Kapturowski S, Ostrovski G, Quan J, Munos R, Dabney W (2018) Recurrent experience replay in distributed reinforcement learning. In: International conference on learning representations. https://openreview.net/forum?id=r1lyTjAqYX

Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu AA, Milan K, Quan J, Ramalho T, Grabska-Barwinska A, et al (2017) Overcoming catastrophic forgetting in neural networks. Proc Natl Acad Sci 114(13):3521–3526

Koenig S, Simmons RG (1993) Complexity analysis of real-time reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence, pp 99–107

Kulkarni TD, Narasimhan K, Saeedi A, Tenenbaum J (2016) Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation. In: Advances in neural information processing systems, pp 3675–3683

Lanctot M, Zambaldi V, Gruslys A, Lazaridou A, Tuyls K, Pérolat J, Silver D, Graepel T (2017) A unified game-theoretic approach to multiagent reinforcement learning. In: Advances in neural information processing systems, pp 4190–4203

Lattimore T, Hutter M, Sunehag P, et al (2013) The sample-complexity of general reinforcement learning. In: Proceedings of the 30th international conference on machine learning

Levine S, Koltun V (2013) Guided policy search. In: International conference on machine learning, pp 1–9

Levine S, Pastor P, Krizhevsky A, Ibarz J, Quillen D (2018) Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. Int J Robot Res 37(4–5):421–436

Madumal P, Miller T, Sonenberg L, Vetere F (2019) Explainable reinforcement learning through a causal lens. arXiv:190510958

Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing Atari with deep reinforcement learning. arXiv:13125602

Mnih V, Badia AP, 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 (ICML), pp 1928–1937

Nagabandi A, Clavera I, Liu S, Fearing RS, Abbeel P, Levine S, Finn C (2018) Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. arXiv:180311347

Nowé A, Vrancx P, De Hauwere YM (2012) Game theory and multi-agent reinforcement learning. In: Reinforcement learning. Springer, Berlin, pp 441–470

Papavassiliou VA, Russell S (1999) Convergence of reinforcement learning with general function approximators. In: International joint conference on artificial intelligence, vol 99, pp 748–755

Pathak D, Agrawal P, Efros AA, Darrell T (2017) Curiosity-driven exploration by self-supervised prediction. In: Proceedings of the international conference on machine learning (ICML)

Peng XB, Andrychowicz M, Zaremba W, Abbeel P (2018) Sim-to-real transfer of robotic control with dynamics randomization. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, Piscataway, pp 1–8

Ramstedt S, Pal C (2019) Real-time reinforcement learning. In: Advances in neural information processing systems, pp 3067–3076

Rusu AA, Rabinowitz NC, Desjardins G, Soyer H, Kirkpatrick J, Kavukcuoglu K, Pascanu R, Hadsell R (2016a) Progressive neural networks. arXiv:160604671

Rusu AA, Vecerik M, Rothörl T, Heess N, Pascanu R, Hadsell R (2016b) Sim-to-real robot learning from pixels with progressive nets. arXiv:161004286

Sadeghi F, Levine S (2016) Cad2rl: Real single-image flight without a single real image. arXiv:161104201

Shoham Y, Powers R, Grenager T (2003) Multi-agent reinforcement learning: a critical survey. Web manuscript

Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, et al (2018a) A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science 362(6419):1140–1144

Silver T, Allen K, Tenenbaum J, Kaelbling L (2018b) Residual policy learning. arXiv:181206298

Song HF, Abdolmaleki A, Springenberg JT, Clark A, Soyer H, Rae JW, Noury S, Ahuja A, Liu S, Tirumala D, et al (2019) V-MPO: On-policy maximum a posteriori policy optimization for discrete and continuous control. arXiv:190912238

Sukhbaatar S, Lin Z, Kostrikov I, Synnaeve G, Szlam A, Fergus R (2018) Intrinsic motivation and automatic curricula via asymmetric self-play. In: International conference on learning representations. https://openreview.net/forum?id=SkT5Yg-RZ

Tan M (1993) Multi-agent reinforcement learning: independent vs. cooperative agents. In: Proceedings of the international conference on machine learning (ICML)

Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P (2017) Domain randomization for transferring deep neural networks from simulation to the real world. In: International conference on intelligent robots and systems (IROS)

Vezhnevets AS, Osindero S, Schaul T, Heess N, Jaderberg M, Silver D, Kavukcuoglu K (2017) Feudal networks for hierarchical reinforcement learning. In: Proceedings of the 34th international conference on machine learning, vol 70, pp 3540–3549. JMLR.org

Vinyals O, Babuschkin I, Czarnecki WM, Mathieu M, Dudzik A, Chung J, Choi DH, Powell R, Ewalds T, Georgiev P, et al (2019) Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575(7782):350–354

Yu W, Tan J, Liu CK, Turk G (2017) Preparing for the unknown: learning a universal policy with online system identification. arXiv:170202453

Zhou W, Pinto L, Gupta A (2019) Environment probing interaction policies. arXiv:190711740