GAN-Based Planning Model in Deep Reinforcement Learning

Song Chen[1], Junpeng Jiang[1], Xiaofang Zhang[1,2(✉)], Jinjin Wu[1], and Gongzheng Lu[2]

[1] School of Computer Science and Technology, Soochow University, Suzhou, China
xfzhang@suda.edu.cn

[2] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

Abstract. Deep reinforcement learning methods have achieved unprecedented success in many high-dimensional and large-scale space sequential decision-making tasks. In these methods, model-based methods rely on planning as their primary component, while model-free methods primarily rely on learning. However, the accuracy of the environmental model has a significant impact on the learned policy. When the model is incorrect, the planning process is likely to compute a suboptimal policy. In order to get a more accurate environmental model, this paper introduces the GAN-based Planning Model (GBPM) exploiting the strong expressive ability of Generative Adversarial Net (GAN), which can learn to simulate the environment from experience and construct implicit planning. The GBPM can be trained using real transfer samples experienced by the agent. Then, the agent can utilize the GBPM to produce simulated experience or trajectories so as to improve the learned policy. The GBPM can act as a role for experience replay so that it can be applied to both model-based and model-free methods, such as Dyna, DQN, ACER, and so on. Experimental results indicate that the GBPM can improve the data efficiency and algorithm performance on Maze and Atari 2600 game domain.

Keywords: Deep reinforcement learning · Model-based · Planning · Generative Adversarial Net

1 Introduction

Reinforcement learning methods [21,22] can be divided into model-based and model-free methods. The use of deep neural networks [8] combined with model-free reinforcement learning methods has made great progress in developing effective agents for a wide range of fields, where the original observations directly map to values or actions. For example, the Deep Q-Network (DQN) [10,11] has tackled many Atari games with complex visual input successfully. However, compared with model-based methods, a model-free approach fails to exploit the underlying sequential nature of decision-making [17], and thus it has no planning.

I. Farkaš et al. (Eds.): ICANN 2020, LNCS 12397, pp. 323–334, 2020.
https://doi.org/10.1007/978-3-030-61616-8_26

Model-based methods [5] rely on planning as their primary component, while model-free methods [4,19] primarily rely on learning. The key idea of model-based methods is to learn a model of the environment and then plan with this model. This learned model is used to evaluate and select among possible policies. As a sequence, model-based methods can support generalization to state samples that are not previously experienced and help to build the connection between present actions and future rewards. The most attractive advantage of model-based methods is that it can improve performance by increasing simulation steps.

In deep reinforcement learning, experience replay (ER) [1,6,10,11] is widely used to reduce sample correlation. Experience replay stores experience tuples which are sampled during training. The agent selects several transitions from the buffer and updates the value function. Experience replay can be viewed as a model-based RL method [13], where the buffer acts as a model of the environment. However, ER does not perform multi-step rollouts of hypothetical trajectories according to a model and it just replays previous agent-environment transitions. As a result, ER can avoid model errors, but it causes bias in updating.

In this paper, we introduce the GAN-based Planning Model (GBPM) exploiting the strong expressive ability of Generative Adversarial Net (GAN) [7], which can learn to simulate the environment from experience and construct implicit planning. The GBPM can get a more accurate model of the environment and it is data-efficient. During training, the GBPM can be trained by using real transfer samples the agent experienced and the agent can utilize the GBPM to produce simulated experiences or trajectories so as to polish the learned policy. An agent that uses the GBPM to implement two ways of planning: one is using simulated experience by the GBPM to improve policy, the other is using simulated experience by the GBPM to select an action for the current state.

The main contributions of this paper are: (1) This paper proposes a novel learnable model-based planning module for simulating the environment in RL. An agent can learn to construct a plan via the GAN-based Planning Model. (2) The GBPM can act as a role for experience replay so that it can be applied to both model-based and model-free methods. This paper integrates the GBPM into some deep reinforcement learning methods effectively, including Dyna [20, 21], DQN [11], and ACER [24]. (3) In this paper, comprehensive experiments are conducted to evaluate the algorithms integrated with GBPM, involving a maze solving problem and Atari 2600 games. The experiment results verify the improvement of algorithm performance in different state dimensions.

2 Related Work

Many effective work has focused on planning with model-based RL. The classic "Dyna" algorithm [21] learns a model of the environment, which is then used to train a policy. Tamar et al proposed the value iteration network (VIN) [23], which is a fully differentiable neural network with a planning module. It trains a deep CNN to plan via iterative rollouts implicitly. Similar to this work, David Silver et al presented the "Predictron" [18], which consists of a fully abstract model,

represented by a Markov reward process, that can be rolled forward multiple "imagined" planning steps. The closest work to "Predictron" is "QMDP-net" [9], which is a neural network architecture for planning under partial observability. It combines the strengths of model-free learning and model-based planning. Similarly, Theophane Weber et al introduced Imagination-Augmented Agents (I2As) [15], which prescribes how a model should be used to arrive at a policy. It learns to interpret predictions from a learned environment model to construct implicit plans in arbitrary, by using the predictions as an additional context in deep policy networks. To prescribe how to construct a plan, Razvan Pascanu et al introduced the "Imagination-based Planner (IBP)" [14]. The IBP can learn to construct, evaluate, and execute plans and it can learn when to act versus when to imagine.

Many model-free reinforcement learning methods rely on experience replay. Experience replay can be viewed as a model-based RL method, where the buffer acts as a model of the environment. To improve the utilization of important samples, Schaul et al proposed a deep reinforcement learning with prioritized experience replay (PER) algorithm [16]. Wang et al presented an actor-critic DRL agent with experience replay (ACER) [24] and it adopts three innovations, including truncated importance sampling with bias correction, stochastic dueling network architectures, and efficient trust region policy optimization. To provide some of the benefits of both Dyna-style planning and ER, Pan et al developed a novel semi-parametric Dyna algorithm [13].

In this paper, with the strong expressive ability of GAN, we aim to propose a model that can learn to simulate the environment from experience and construct implicit planning. Furthermore, this model can act as a role for experience replay so that it can be applied to both model-based and model-free methods.

3 The GBPM Architecture

The GBPM is a model-based agent which learns from previous experience and can make effective planning about the future. We can get a more accurate model of the environment from the GBPM and it is data-efficient.

3.1 The GBPM Module

The architecture of the GBPM is shown in Fig. 1. The GBPM module is made up of GAN and it consists of a deep generative network model called Generator (G) and a deep discriminative network model called Discriminator (D). The Generator (G) and the Discriminator (D) are both deep network models.

In our work, the simulated model of the environment which is learned by the GBPM is used to predict the next state and the reward signals from the environment. In detail, the transition samples that agent experiences are considered as the training samples of the GBPM. The form of the transition samples is (s_t, a_t, r_t, s_{t+1}). The GBPM takes the tuple of (s_t, a_t) of transition samples as the noise input of the Generator (G). While the input of the Discriminator

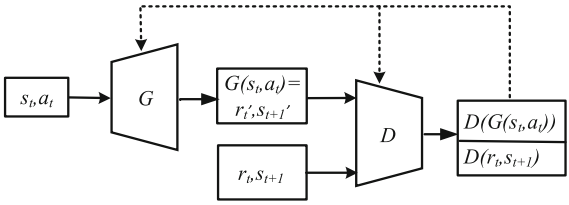


Fig. 1. The architecture of the GBPM.

(D) is the tuple of (r_t, s_{t+1}), that is the true reward and next state. As a result, the Generator (G) outputs the predicted reward and the next state according to the state and action in the time step t. This process can be described as $G(s_t, a_t) = r_t', s_{t+1}'$. The Discriminator (D) outputs the probability of discriminating the newly generated transition sample that comes from the real sample.

During training, the generator and the discriminator take constantly confrontational training according to the actual transition samples. Thus, the GBPM can fit the environment better and implicit more accurately planning. Then the agent can take advantage of it to produce simulated experience or trajectories with the purpose of improving the learned policy.

The GBPM has an excellent ability for generalization. It can make accurate predictions of the future of agent. These generative samples may have been experienced by the agent, or perhaps the agent has never experienced. Therefore, the GBPM can effectively plan and guide the agent to improve its policy. In addition, the GBPM can replace the experience replay in some model-free methods. So it can be applied to both model-based and model-free methods.

3.2 Dyna with GBPM

Dyna is one of the classic model-based reinforcement learning algorithms. It is an integrated architecture for learning, planning and reacting. It is specifically designed for the situation in which the agent does not have complete knowledge of the effects of its actions on the environment.

Within a planning agent, real experience has at least two roles: it can be used to improve the model (to make it more accurately match the real environment), it is called model-learning. In our work, we use real experience to improve the GBPM. Real experience can also be used to directly improve the value function and policy using different kinds of reinforcement learning methods, such as Q-learning. In this case, it is called direct reinforcement learning.

DynaQ with the GBPM is a simple architecture integrating the major functions needed in an online planning agent. The overall architecture of DynaQ with the GBPM is shown in Fig. 2. The planning methods and the direct RL methods are both Q-learning. The agent interacts with the environment and produces real experience. We use it to train the GBPM and the GBPM can simulate the experience for planning.

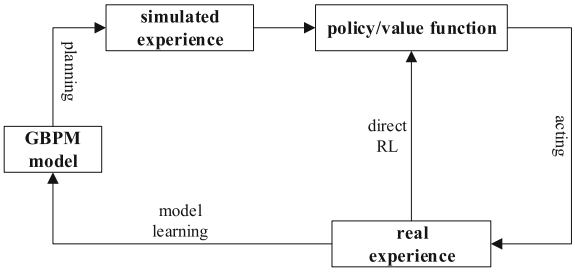


Fig. 2. The architecture of DynaQ with the GBPM.

3.3 DQN with GBPM

In order to improve the performance of the DQN on some tasks, the GBPM planning module is embedded to make DQN obtain the ability of planning.

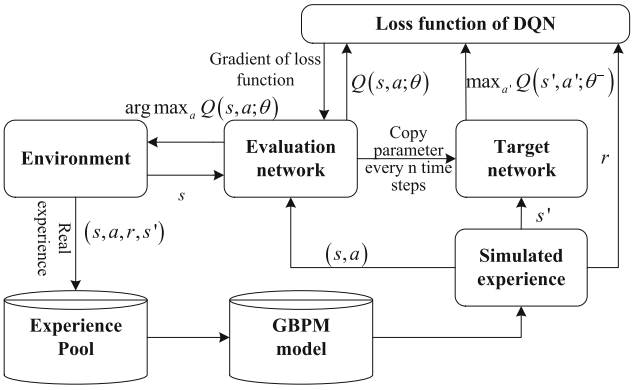


Fig. 3. The architecture of the DQN with the GBPM.

The complete training process of DQN with the GBPM is shown in Fig. 3. First of all, traditional DQN uses the experience replay mechanism. In our work, we replace the experience replay with the GBPM and still retain the use of the experience pool. We do not use the real experience to train the network but to learn a model of the environment with GAN, and it can produce simulated experience to train the network for planning. At each time step, the transferred samples $e_t = (s_t, a_t, r_t, s_{t+1})$ obtained by interacting the agent with the environment are stored in the experience pool $D = \{e_1, ..., e_t\}$. During training, each time a fixed number of transfer samples (mini-batch) are randomly selected from the experience pool to train the GBPM. Then the GBPM produces mini-batch simulated transition samples to update the parameters θ of the network with the Stochastic Gradient Descent algorithm.

3.4 ACER with GBPM

ACER is an actor-critic deep reinforcement learning algorithm with experience replay. Importance sampling is applied to control the variance and stability in ACER. It retrieves a trajectory $\{s_0, a_0, r_0, \mu(\cdot|s_0), ..., s_k, a_k, r_k, \mu(\cdot|s_k)\}$, where the actions have been sampled according to the behavior policy μ, from the memory of experience.

While in our work, ACER with the GBPM first samples a transition trajectory from the experience pool to train the GBPM and then it simulates a new trajectory according to the behavior policy μ. So the importance weighted policy gradient can be calculated by the simulated trajectory.

The rest of the ACER with the GBPM algorithm is exactly the same as ACER [24]. It estimates the action value function $Q^{\pi}(s_t, a_t)$ using Retrace [12] with multi-step estimation. To avoid high variance, importance weight truncation with bias correction is adopted in ACER. Moreover, ACER uses an efficient trust region policy optimization method which maintains an average policy network that represents a running average of past policies. In this paper, we embed a trainable planning module with GAN which can produce a simulated trajectory for training the network in the ACER.

4 Experiment

In this section, the performance of DynaQ and DynaQ-GBPM is compared on the Maze domain which is a simple maze that contains some obstacles [21]. Then the performance of DQN-GBPM and ACER-GBPM are evaluated on Atari games domain which is a discrete action domain.

4.1 Results on Maze Domain

Simple Maze. Our first experiment domain is a simple maze with some obstacles shown in Fig. 4. Each square can be considered as a state. The square with "S" is the start state and the square with "G" is the goal state.

In each state, there are four actions: up, down, right, and left, which take the agent deterministically to the corresponding neighboring states. When the agent moves to an obstacle or the edge of the maze, the agent remains where it is. The reward is zero on all transition, except those into the goal state, on which it is +1. After reaching the goal state, the agent returns to the start state to begin a new episode. This is a discounted and episodic task with $\gamma = 0.95$.

In this section, we first carry out an experiment in which DynaQ and DynaQ-GBPM agents are applied to the simple maze shown in Fig. 4. The average learning curves for the simple maze are shown in Fig. 5. The generator and the discriminator in the GBPM are both deep neural networks. The initial action values are zero, the step-size parameter is $\alpha = 0.1$ and the exploration parameter is $\epsilon = 0.1$. The planning step n of DynaQ and DynaQ-GBPM is 5. The curves show the number of steps taken by the agent to reach the goal in every episode, averaged over 30 repetitions of the environment.

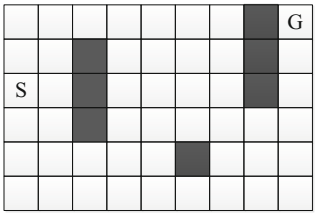


Fig. 4. The simple Maze domain.

Figure 5 shows that DynaQ-GBPM finds the solution to the goal faster than DynaQ. In DynaQ and DynaQ-GBPM, learning and planning are accomplished by exactly the same algorithm, operating on real experience for learning and on simulated experience for planning. However, the GBPM can simulate more accurate samples for the agent and can learn to fit the environment better. As the new experience is gained, the model is updated to match reality better. As the model changes, the planning process in DynaQ-GBPM will gradually compute a different way of behaving to match the new model.

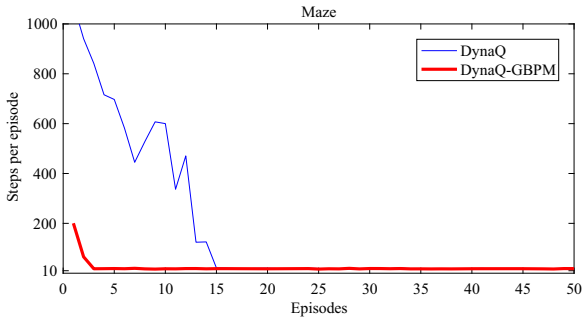
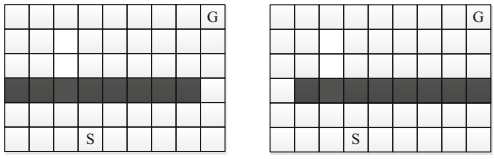


Fig. 5. Comparisons of DynaQ and DynaQ-GBPM for Simple Maze domain.

Blocking Maze. The second experiment domain is a blocking maze, which can be used to illustrate the existence of model error and then to check whether the agent can recover from the wrong model. The blocking maze is shown in Fig. 6. In Fig. 6(a), we can see that there is a short path from "S" to "G". Then after 1000 time steps, the position of the barrier has changed and the short path is "blocked". So we need to take a longer path along the left of the barrier from "S" to "G" which is shown in Fig. 6(b).

The graph of the average reward for DynaQ and DynaQ-GBPM is shown in Fig. 7. Both two agents find the short path within 1000 time steps, and the DynaQ-GBPM agent performs better than the DynaQ agent. After 1000 time



(a) Before 1000 steps (b) After 1000 steps

Fig. 6. The Blocking Maze domain.

steps, the environment has changed. In this period, these two agents obtain no reward because they are wandering around behind the barrier. After a period of exploration, they are able to find the shortest path for a new optimal policy. The DynaQ-GBPM agent still performs better than the DynaQ agent and the gap between them gradually grows with increasing time step. So we can conclude that the DynaQ-GBPM agent easily understands the changes in the environment and can recover from the wrong model.

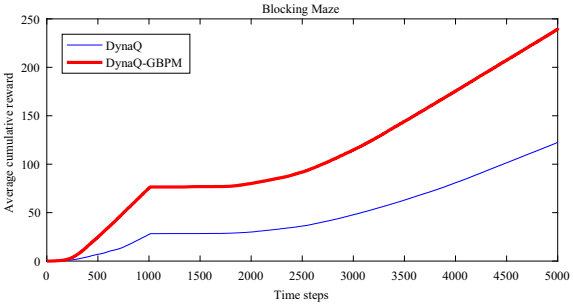


Fig. 7. Comparisons of DynaQ and DynaQ-GBPM for Blocking Maze domain.

4.2 Results on Atari Games Domain

The second experiment domain is the Atari 2600 game environment in the OpenAI Gym [2]. In this section, We compare the performance of DQN and DQN-GBPM, ACER and ACER-GBPM on four strategic Atari 2600 games.

DQN and DQN-GBPM. To compare the performance of DQN and DQN-GBPM, we use the same set of parameters in DQN [11]. The value network structure of DQN-GBPM is the same as DQN. The value network consists of three convolutional layers and two fully connected layers. The network architecture of the GBPM in DQN-GBPM is exactly the same as in infoGAN [3]. The generator consists of two fully connected layers and two convolutional layers. The

discriminator is made up of two convolutional layers and two fully connected layers. The experiment conducts 200 independent training periods (Epoch) which includes 200000 steps parameter updating for each game.

We first compare the rewards of DQN, DQN-PER [16] and DQN-GBPM during each epoch of training the agent to play Atari 2600 games, including Seaquest, Amidar, Gravitar, and Alien. These four games have been widely used in recent related works. The DQN algorithm uses the normal experience replay which takes the samples from the experience pool with moderate probability. While the DQN-PER makes use of the prioritized experience replay which uses the time difference error of each sample as the criterion for evaluating the priority of samples. Different from these, the DQN-GBPM algorithm applies the GBPM to produce simulated experience.

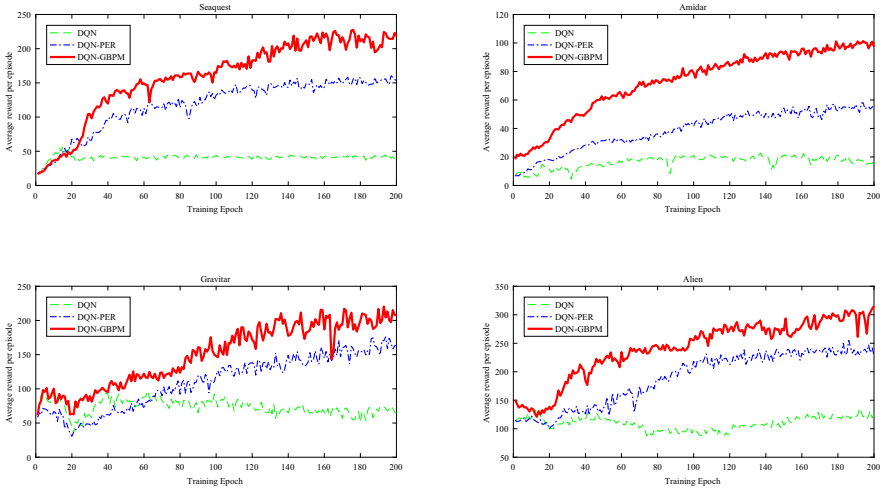


Fig. 8. Comparisons of DQN, DQN-PER and DQN-GBPM for Atari games

The results are shown in Fig. 8. It is indicated that DQN-GBPM outperforms DQN and DQN-PER on all of four Atari 2600 games. The GBPM can learn the real model of the environment based on the experience of the agent, so the DQN-GBPM algorithm gains the ability of planning and it can generalize to many states that the agent has never experienced. By contrast, the normal experience replay and the prioritized experience replay cannot express the environment model well, the performance of the DQN and DQN-PER algorithm is slightly worse than that of the DQN-GBPM algorithm.

ACER and ACER-GBPM. Most of the parameters used in the ACER and ACER-GBPM algorithms are identical in the experiment. The parameter settings are consistent with the parameter settings in the ACER algorithm [24].

When using experience replay, ACER gives each thread an experience pool with a capacity of 50000 samples. Moreover, ACER just uses real experience in the replay memory to generate sample trajectories for training. While ACER-GBPM produces a simulated sample trajectory for training using the GBPM environment model learned from real sample experience in the experience pool. The network architecture of the GBPM in ACER-GBPM is exactly the same as in DQN-GBPM. The experiment conducts 1,000 training periods (Epoch) and each Epoch includes 80,000 steps. Thus, a total of 80,000,000 steps are trained.

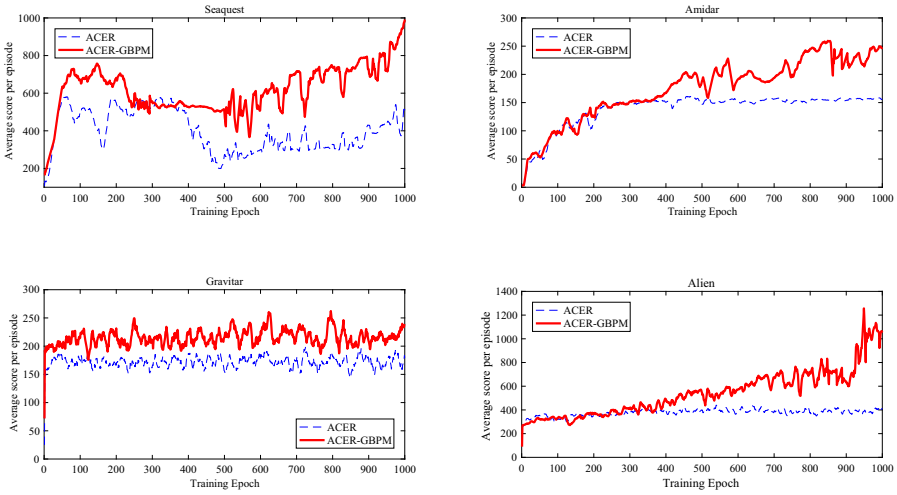


Fig. 9. Comparisons of ACER and ACER-GBPM for Atari games

We also compared the rewards of ACER and ACER-GBPM during each epoch of training the agent to play Atari 2600 games, including Seaquest, Amidar, Gravitar, and Alien. The results are shown in Fig. 9. ACER-GBPM outperforms ACER on all of four Atari 2600 games. Thus, we can conclude that the GBPM can learn the real model of the environment based on the experience of the agent and using the GBPM to produce simulated sample trajectories for training can improve the performance of the ACER algorithm.

5 Conclusion

In this paper, we have introduced the GAN-based Planning Model exploiting the strong expressive ability of GAN, which can learn to simulate the environment from experience and construct implicit planning. We have integrated the GBPM into some deep reinforcement learning methods effectively, like Dyna, DQN and ACER, endowing them the ability to learn model-based planning from GAN. A series of experiments on Maze and Atari 2600 game domains are conducted and

empirical experiment results show that the GBPM does improve the performance of these algorithms.

In future work, how to improve the GBPM to make it more generalizable needs to be considered. Furthermore, how to embed the GBPM into other deep reinforcement learning algorithms based on planning, like Monte Carlo Tree Search is an interesting topic.

Acknowledgment. This work was supported in part by National Natural Science Foundation of China (61772263), Suzhou Technology Development Plan (SYG201807), and the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

1. Adam, S., Busoniu, L., Babuska, R.: Experience replay for real-time reinforcement learning control. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **42**(2), 201–212 (2011). https://doi.org/10.1109/TSMCC.2011.2106494
2. Brockman, G., et al.: OpenAI gym. arXiv preprint arXiv:1606.01540 (2016)
3. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2172–2180 (2016)
4. Degris, T., Pilarski, P.M., Sutton, R.S.: Model-free reinforcement learning with continuous action in practice. In: 2012 American Control Conference (ACC), pp. 2177–2182. IEEE (2012). https://doi.org/10.1109/ACC.2012.6315022
5. Doya, K., Samejima, K., Katagiri, K.I., Kawato, M.: Multiple model-based reinforcement learning. Neural Comput. **14**(6), 1347–1369 (2002). https://doi.org/10.1162/089976602753712972
6. Foerster, J., et al.: Stabilising experience replay for deep multi-agent reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 1146–1155. JMLR.org (2017)
7. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
8. Hinton, G., et al.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Sig. Process. Mag. **29**(6), 82–97 (2012). https://doi.org/10.1109/MSP.2012.2205597
9. Karkus, P., Hsu, D., Lee, W.S.: QMDP-Net: deep learning for planning under partial observability. In: Advances in Neural Information Processing Systems, pp. 4694–4704 (2017)
10. Mnih, V., et al.: Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
11. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
12. Munos, R., Stepleton, T., Harutyunyan, A., Bellemare, M.: Safe and efficient off-policy reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 1054–1062 (2016)
13. Pan, Y., Zaheer, M., White, A., Patterson, A., White, M.: Organizing experience: a deeper look at replay mechanisms for sample-based planning in continuous state domains. arXiv preprint arXiv:1806.04624 (2018). https://doi.org/10.24963/ijcai.2018/666

14. Pascanu, R., et al.: Learning model-based planning from scratch. arXiv preprint arXiv:1707.06170 (2017)
15. Racanière, S., et al.: Imagination-augmented agents for deep reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 5690–5701 (2017)
16. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
17. Shadlen, M.N., Shohamy, D.: Decision making and sequential sampling from memory. Neuron **90**(5), 927–939 (2016). https://doi.org/10.1016/j.neuron.2016.04.036
18. Silver, D., et al.: The predictron: end-to-end learning and planning. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 3191–3199. JMLR.org (2017)
19. Strehl, A.L., Li, L., Wiewiora, E., Langford, J., Littman, M.L.: PAC model-free reinforcement learning. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 881–888 (2006). https://doi.org/10.1145/1143844.1143955
20. Sutton, R.S.: Dyna, an integrated architecture for learning, planning, and reacting. ACM SIGART Bull. **2**(4), 160–163 (1991). https://doi.org/10.1145/122344.122377
21. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
22. Sutton, R.S., Barto, A.G., et al.: Introduction to Reinforcement Learning, vol. 135. MIT Press, Cambridge (1998)
23. Tamar, A., Wu, Y., Thomas, G., Levine, S., Abbeel, P.: Value iteration networks. In: Advances in Neural Information Processing Systems, pp. 2154–2162 (2016)
24. Wang, Z., et al.: Sample efficient actor-critic with experience replay. arXiv preprint arXiv:1611.01224 (2016)