



Reinforcement Learning Policy with Proportional-Integral Control

Ye Huang, Chaochen Gu^(✉), Kaijie Wu, and Xinping Guan

Key Laboratory of System Control and Information Processing, MOE of China,
Shanghai Jiao Tong University, Shanghai 200240, China
{lutein,jacygu,kaijiewu,xpguan}@sjtu.edu.cn

Abstract. Deep Reinforcement Learning has made impressive advances in sequential decision making problems recently. Constructive reinforcement learning (RL) algorithms have been proposed to focus on the policy optimization process, while further research on different network architectures of the policy has not been fully explored. MLPs, LSTMs and linear layer are complementary in their controlling capabilities, as MLPs are appropriate for global control, LSTMs are able to exploit history information and linear layer is good at stabilizing system dynamics. In this paper, we propose a “Proportional-Integral” (PI) neural network architecture that could be easily combined with popular optimization algorithms. This PI-patterned policy network obtains the advantages of integral control and linear control that are widely applied in classic control systems, improving the sample efficiency and training performance on most RL tasks. Experimental results on public RL simulation platforms demonstrate the proposed architecture could achieve better performance than generally used MLP and other existing applied models.

Keywords: Reinforcement learning · Deep learning · Neural network
Control theory

1 Introduction

Recently, Deep Reinforcement Learning (DRL) has made notable advances in solving representative benchmark problems, especially in simulated control [11, 22], continuous robot control [5, 9, 13], Go game [24], Atari games [15] and other sequential decision making domains. Directing an agent to interact with the environment, the policy network of DRL is of critical importance to achieve maximum cumulative long time reward. Generally, Convolutional Neural Network (CNN) is applied in visual tasks such as high-dimensional control of robots that utilizes raw visual images or videos as input. As for non-visual tasks, the widely-used Multi-Layer Perceptron (MLP) is considered as a basic policy network structure for many DRL algorithms. However, inductive research on the effectiveness of policy network architecture remains to be further explored. It's necessary to draw importance on the policy architecture to improve agent's performance better.

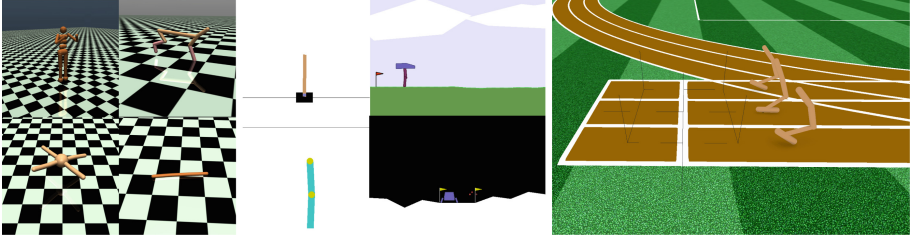


Fig. 1. Several popular reinforcement learning tasks implemented in public simulation platforms MuJoCo, OpenAI Gym and Roboschool. Including continuous control of simulated robots, classical control problems and games, etc.

In this work, we present an effective policy network architecture that is generic in handling benchmark RL problems from board games to simulated control tasks. Inspired by the Proportional-Integral Controller widely used in practical control systems, we introduce the memory mechanism of Long Short-Term Memory (LSTM) into policy network, in which characteristic could be found a clue to lead the agent exploits history information implicitly. With LSTM functioning as the integral controller, the “proportional” part is modelled as the linear projection of inputs. To better stabilize system dynamics, we use non-linear controller additionally. It’s convenient to combine the proposed network with many existing DRL algorithms. The consolidation of linear, nonlinear and “integral” controllers could enhance the robustness and generalization of policy network compared with the typically applied MLP structure. Given current state, these three branches would evaluate respectively and then their results are combined to compute the final action.

Compared with generally applied MLPs policy networks, our history-concerned PI architecture could improve the performance of model on various RL tasks, especially on continuous control tasks. The key insight of our work is that the combination of control, memory mechanism and deep learning has distinct influence on the training efficiency and generalization ability. To validate the effectiveness of this policy network, extensive experiments are conducted on both classic control tasks as well as complex sequential decision making problems, such as pendulum control and humanoid walking, which are wrapped as standard RL environments in public simulation platforms such as MuJoCo [16], OpenAI Gym [3], Roboschool. We further perform different ablation experiments utilizing different policy optimization algorithms like Deep Deterministic Policy Gradient (DDPG) [11], Proximal Policy Optimization (PPO) [22], Actor Critic using Kronecker-Factored Trust Region (ACKTR) [31], etc. Our experimental results demonstrate that the proposed architecture is capable of enhancing model generalization as well as training efficiency compared with existing works.

In our paper, Sect. 2 introduces relevant researches about classic RL optimization methods and generally used network architectures, as well as the embedding of LSTM. In Sect. 3, we explain the proposed architecture in details and analyze

its theoretical applicability. Experiment results are shown in Sect. 4, with several ablation experiments involved.

2 Related Works

Reinforcement learning problems are basically formulated as Markov Decision Process (MDP), in which an agent interacts with dynamic environment through trial-and-error [7]. Focusing on goal-directed learning, DRL algorithms are proved to be applicable for many sequential decision making problems in robotics [5, 9, 13], video games [15], simulations [11, 21] and even self-driving systems [23]. Traditional approaches such as dynamic programming [10] and control methods fail to solve these challenges since the delayed feedback, unknown environment dynamics and the curse of high dimensions [7].

Constructive works on RL training process have been proposed in recent years. Generally, there are three main branches in RL, value-based, policy-based and actor-critic (combine both) methods. While classic value-based approaches such as SARSA [18] and Q-learning [28] have been shown unable to converge to policy for simple MDPs [26], recent model-free DRL algorithms have made dramatic advances in solving continuous control tasks. DQN [15] proposed by Google DeepMind has attracted great interest in the machine learning community, and for stochastic policy optimization, other policy-based algorithms such as Trust Region Policy Optimization (TRPO) [20], PPO [22], Asynchronous Advantage Actor-critic (A3C) [14], are effective in training the agents for accumulating more rewards through time. Policy gradient optimization methods only utilize states input for end-to-end training without any prior information.

However, further exploiting the applicability of different network architectures has not been fully studied. Most of the methods we discuss above adopt standard neural networks like MLPs, single LSTMs or autoencoders as policy network for the non-vision part, and pay their attention to optimization algorithms. Few works focus on using the internal structure in the policy parameterization to speed up learning process [30] and adding inductive bias to policy networks [27]. [27] proposed a novel network architecture named dueling architecture that represents separate estimators for state value function and state-dependent action advantage function respectively. Though splitting the Q-network into two streams, the Dueling Network can't deal with many continuous control tasks.

Similar work [17] proposes two applicable policy architectures: linear policy that maps from observations to actions, RBF policy that uses random Fourier features of the observations. These two architectures can achieve relatively promising performance on some continuous control tasks while still lacks generalization for most RL problems. Work [25] demonstrates that linear policy could make a complement to standard MLP network and this combination policy improves the sample efficiency, episodic reward and robustness. These relevant researches prove that the integration of linear and other specific architectures has potential for generating more effective models.

As a class of Recurrent Neural Networks (RNNs) architecture, LSTM is designed to learn temporal sequences and the long-term dependencies [12]. [2] presents a model-free RL-LSTM framework to solve non-Markovian tasks, and [29] uses LSTM to train an end-to-end dialog systems that is optimized with supervised learning and reinforcement learning. The inspirational application of LSTM in RL problems demonstrate that LSTM is capable of processing internal state information and exploring the long-term dependency between relevant events in other benchmark RL problems.

The idea of integrating LSTM with linear network and standard MLP network could be much similar to the traditional feedback control approach PI [1] which has been successfully applied to a variety of continuous control like robotics, unmanned air vehicles [19] and other automatic systems. Inspired by the widely used PI controller, we propose the “Proportional-Integral” policy network to represent the physical interpretations of control approach. Our architecture is easily to be combined with existing RL optimization algorithms and sufficient experiments show this policy network could achieve remarkable results on many benchmark tasks outperforming the results achieved by similar works [25].

3 Approach

3.1 Background

In the process of optimizing episodic reward while interacting with dynamic environment, the agent updates the policy π according to Bellman (Optimality) Equation. We formulate the standard RL environment of a sequential decision making problem as Markov Decision Process (MDP) defined by the tuple: $\mathcal{M} = \{\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma\}$, in which $\mathcal{S} \subseteq \mathbb{R}^n$ is an n-dimensional state space, \mathcal{O} the observation space, $\mathcal{A} \subseteq \mathbb{R}^m$ an m-dimensional action space, \mathcal{R} a bounded reward function, \mathcal{P} a transition probability function, and $\gamma \in (0, 1]$ a discount factor.

At every time step t , the agent is given current state $s_t \in \mathcal{S}$ or observation $o_t \in \mathcal{O}$ and chooses one action a_t from finite action set \mathcal{A} according to the policy $\pi_\theta(a_t|s_t)$ parameterized by θ . In problems with visual inputs, observation o_t is directly obtained from the environment and then processed by convolutional neural network to be fed into policy network. The performed action would affect the subsequent state iteratively because after action taken, the environment would return a reward value r and then transit to the next state s_{t+1} according to state transition probability matrix $\mathcal{P} = P(s_{t+1}|s_t, a_t)$. For example, in Atari domain, the player agent perceives current video as observation information, then chooses an action to perform and receives reward signal returned by game emulator.

The goal of RL is to learn an optimal policy that maximizes the total discounted reward through trading-off the exploration and exploitation. G_t is defined as the sum of discounted reward from time-step t :

$$\begin{aligned}
G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \\
&= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}
\end{aligned} \tag{1}$$

where discount factor γ determines the present value of future rewards and values immediate reward above delayed reward, reward R at each time-step is a numerical number given by the environment.

We use state value function $V(s)$ to evaluate the long-term value of state s and action-state value function $Q(s, a)$ to figure out the value of state-action pair (s, a) .

$$\begin{aligned}
V^\pi(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t = s]
\end{aligned} \tag{2}$$

$$\begin{aligned}
Q^\pi(s, a) &= \mathbb{E}[G_t | S_t = s, A_t = a, \pi] \\
&= \mathbb{E}[R_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) | S_t = s, A_t = a]
\end{aligned} \tag{3}$$

where policy π is parameterized by θ and experimentally implemented by neural networks.

3.2 Architecture

To apply accurate and optimal control, in this section we present a novel policy network architecture consisting of three independent branches: LSTM for exploiting hidden history information, nonlinear network for global control and linear network for stabilizing the system dynamics. The architecture of the proposed policy network is shown in Fig. 2.

Inspired by the Proportional-Integral Controller (PI) widely used in practical control systems, LSTM is adopted to utilize long-term encoded state information to control action at current time-step, which is similar to the control of the historic cumulative value of error used in PI. The aim of introducing control prior to policy is to eliminate the residual error in training process. Linear control policy has been proved effective for particular RL problems. In addition, we use basic MLP as nonlinear control network for its capability of global control and the promising performance on generic policy networks.

Given current state, three branches of policy $\pi_\theta(a_t | s_t)$ would evaluate respectively and then the results are combined to compute the resulting action at time t :

$$a_t = a_t^l + a_t^n + a_t^r \tag{4}$$

where a_t^l is the output of linear control network, a_t^n is the result of nonlinear policy module and a_t^r the time-dependent LSTM branch.

The key insight of our architecture is that the classic control prior knowledge combined with reinforcement learning has functioned practically well on continuous control tasks. To analyze the theoretical feasibility, we illustrate the generic task as traditional control problem. Let the desired current state denoted as s_t^d and the actual state at time step t as s_t , so the temporary error would be

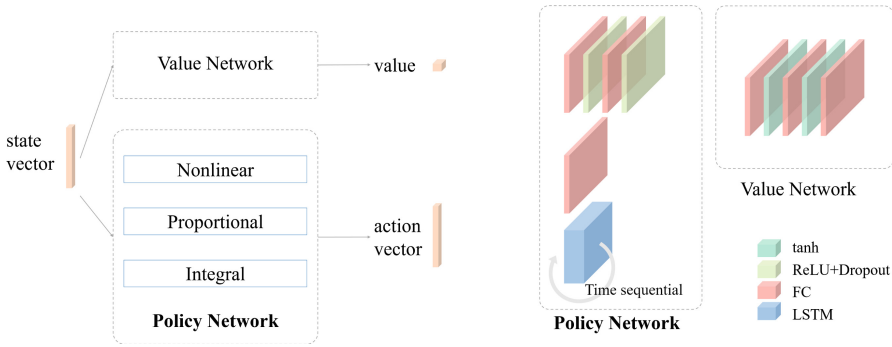


Fig. 2. The pipeline of reinforcement learning and the architecture of the proposed PI policy network.

$e_t = s_t - s_t^d$. According to control theory, the goal of control is to eliminate the error as much as possible.

In this formulation, given current state, the action should be:

$$\begin{aligned}
 a_t &= f_t^r + f_t^s(s_t, s_t^d) + f_t^e \\
 &= f_t^r + f_t^s(s_t, s_t^d) + K_p \cdot (s_t - s_t^d)
 \end{aligned}
 \tag{5}$$

where f_t^r is a history-concerned control term similar to the integral module in PI controller, and f_t^s is the nonlinear control branch formulated as the function of current state and desired state, f_t^e is the function of current error. As we stated before, this error function serves as a linear control module with K_p being the proportional terms for error e_t . In classic control theory, the proportional module is used for removing the gross error by applying the difference between the desired state and the measured state proportionally to the controlled variables. Furthermore, the nonlinear branch f_t^s works as global feedback control based on the predicted environment state s_t^d . “Integral” module f_t^r is utilized to eliminate the residual offset error by taking history error into account.

We further decompose the equation into:

$$\begin{aligned}
 a_t &= f_t^r + f_t^s(s_t, s_t^d) + K_p \cdot (s_t - s_t^d) \\
 &= f_t^r + f_t^s + K_p \cdot s_t - K_p \cdot s_t^d \\
 &= f_t^r + f_t^n + f_t^l
 \end{aligned}
 \tag{6}$$

where we apply the transformation $f_t^n = f_t^s - K_p \cdot s_t^d$. The final control equation is totally the same as Eq. (4) we propose previously where f_t^l is denoted as linear control branch $K_p \cdot s_t$.

Experiment results demonstrate that both linear and nonlinear policy could achieve promising performance, as shown in Fig. 3. On some specific RL tasks like Humanoid, linear policy could obtain comparable effect with baseline MLP network while on more tasks, it fails to perform sound results. The linear control

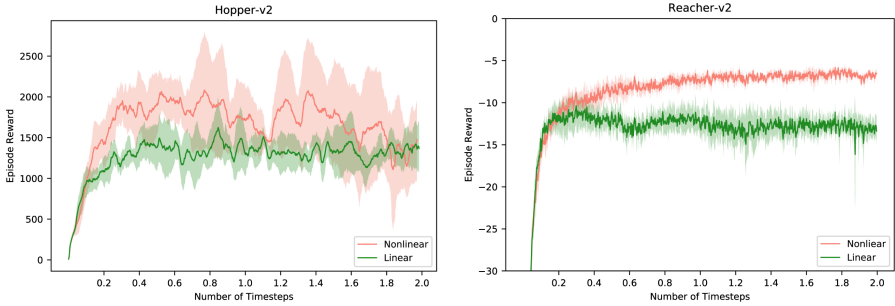


Fig. 3. Averaged learning curves of linear and nonlinear policy network of 5 sets of random seeds.

module f_t^l is implemented as 1 linear layer $K_p \cdot s_t + b$ where the gain matrix K_p and bias b are hyper-parameters need to be learned.

As part of our policy network, LSTM stores states information from previous time steps. This concept of holding long-term encoded information to control current action is similar to control the historic cumulative value of error used in PI, while the specific implementation and practical implication are quite distinct in some degree. In MDP, immediate states would play greater roles than delayed ones, which is in accordance with the internal states of LSTM. That’s why we adopt LSTM as the most effective component of our policy network. In the proposed policy network, we use 1 LSTM layer with 64 hidden states to operate sequential data. The results of PI policy with different number of LSTM layer and hidden states are shown in experiment section.

The nonlinear network is implemented as generic MLP, and we also give the experimentally result that single nonlinear policy could acquire. Generally the individual nonlinear network works effectively for most of the RL tasks, which confirms the necessity of combing this nonlinear policy with the other two streams to further exploit.

4 Experiment

We conduct sufficient experiments on various benchmark RL tasks and widely used simulation environments to validate the applicability and effectiveness of the proposed policy network architecture. We mainly compare the training results with generic MLP policy and similar SCN policy network proposed by [25] under the same conditions. In addition, ablation experiments about the three policy architecture individually and the complement network are performed to confirm the capability. All these experiments are conducted under the guidance of RL reproducibility study [6]. In this section, experiment details and results are clearly explained.

4.1 RL Environments

Generally used RL environments including OpenAI Gym, MuJoCo, Roboschool that contain diverse RL tasks such as Atari games, continuous robot control and classic control problems are shown in Fig. 1. These simulation platforms are built with different physics engines and parameters, thus we could perform adequate validation experiments on available tests as many as possible.

Some standard test environments such as Humanoid-v1 and Swimmer-v1 are implemented in both MuJoCo and Gym. For example, Humanoid-v1 makes a three-dimensional bipedal robot walk forward without falling over. The state of this task is a 47 dimensional vector containing the position and velocity information. The action consists of a discrete 17 dimensional torque control vector over every joint of the humanoid robot.

4.2 Experimental Setup

As indicated before, we apply the proposed policy architecture to baseline RL algorithms PPO, ACKTR, A3C on popular benchmark tasks that have been widely used in the study of DRL. The test tasks consist of complicated continuous control problems, simplified classical control problems as well as Atari games. We mainly use the HalfCheetah-v2 and Hopper-v2 implemented in MuJoCo for their stable and contrasting dynamics.

The comparison experiments are a series of policy networks trained from scratch, including our control network, generic multilayer perceptron (MLP) and Structured Control Net (SCN). To avoid bias, these three policy networks are trained using the same algorithms with fixed hyper-parameters during the training. Applicable training algorithms PPO, ACKTR are implemented from OpenAI Baselines [4]. PPO is optimized by Adam optimizer [8] with initial learning rate as $3e-4$, ϵ term as $1e-8$. Particularly, in PPO generalized advantage estimation is used with $\tau = 0.95$ and the clip parameters is 0.2. In addition, ACKTR uses KFAC optimizer proposed by [31], and the learning rate is $3e-4$, momentum parameter is 0.9.

In order to confirm the fairness, all the experiments we conduct use the same set of random seeds, and the depicted learning curves are obtained by averaging the evaluation results over five different random seeds from 1 to 5 respectively. We train these networks for 2M timesteps over every tasks, and the mini-batch size is fixed to 32, with fine-tuned reinforcement learning parameter discount factor $\lambda = 0.99$. We implement these experiments in PyTorch using 12 cores with Nvidia GeForce GTX 1060.

4.3 Results

In this section, we test three models: the proposed PI policy network, a baseline MLP and Structured Control Network (SCN) to compare their performances. The evaluation metrics widely applied in the reinforcement learning studies

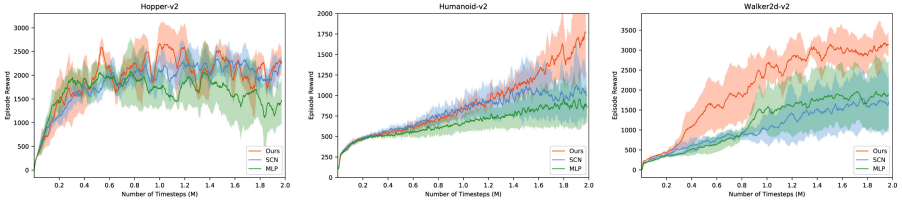


Fig. 4. Episode reward learning curves of the comparative methods: PI network, generic MLP and SCN, averaged on five sets of random seeds.

include the learning curves of cumulative reward along timesteps, the maximum reward and average reward over a fixed number of timesteps (Fig. 4).

The architecture of MLP used in these comparative experiments is a fully connected layer with two hidden layers, each of which consists 64 units and is activated by tanh function. This standard MLP-64 architecture is generally used in many algorithms [22,31]. According to the experiment details described in [25], the SCN is implemented as the combination of a generic three layer MLP (remove the bias of the last linear layer) and a linear layer. We adopt tanh as the activation function for the MLP used in SCN, and the two hidden layers of MLP have 64 units respectively. As shown in Fig. 2, the proposed architecture contains a LSTM with hidden size 64 and two nonlinear layers attached to the end of LSTM, a generic MLP-16 nonlinear network, and a simple linear network. The number of parameters of these three policy networks are quite approaching. For fairness comparison, the value network for all these optimization algorithms and models is fixed to be a three layer nonlinear network with one output dimension.

Comparison of Performance: The learning curves of the episode rewards are shown in Fig. 3. More accurately, the average reward and final episode reward are chosen to represent the model performance, as presented in Table 1. We only show the PPO training results since the ACKTR results are quite similar.

We compute the improvement of average episode reward to depict the training efficiency. In this evaluation, our model achieved 122% averaged reward improvement compared to generic MLP, and 126% to SCN.

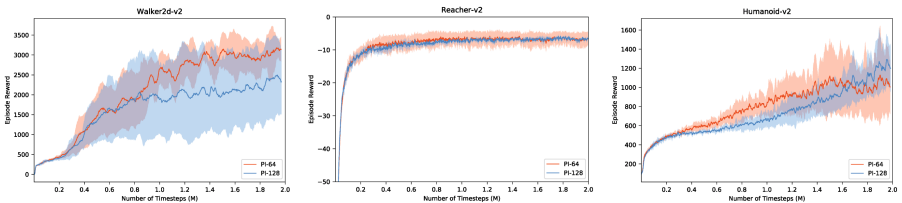


Fig. 5. Episode reward learning curves of the comparative methods: 1 LSTM layer with 64 hidden size PI and 1 LSTM layer with 128 hidden size PI, averaged on five sets of random seeds.

Table 1. Average and final episode rewards on various RL tasks of our model, SCN and MLP model.

Task	Average reward			Final reward		
	Ours	SCN	MLP	Ours	SCN	MLP
Ant	256	370	489	984	1205	1583
HalfCheetah	1395	1129	1390	2356	1181	2277
Hopper	1961	1854	1646	2274	2272	1449
Humanoid	892	789	662	1762	1007	866
Reacher	-9.0	-10.9	-9.3	-6.7	-7.3	-6.6
Swimmer	28	29	34	37	33	49
Walker2d	2075	1052	1191	3133	1678	1895

4.4 Ablation Experiments

To test the comparative effectiveness of the three modules of PI network respectively, we conduct ablation experiments that test each module separated from a fully trained PI model. We compare these branches' performances with an independently trained linear policy and a nonlinear policy (MLP) with the same size of linear and nonlinear branches in PI architecture. The training curves of single linear and nonlinear policy have been shown in Fig. 3. Since branches of PI policy network are jointly trained, we only compare the test rewards. The result of linear network contained in the learned PI model outperforms effectively in simple control tasks such as Pendulum and Walker2d when compared with single trained linear network, and trained MLP branch achieves slightly better rewards compared to separately trained MLP.

We also use LSTM with different number of hidden states and layers to evaluate the effect of proportional branch. As show in Fig. 5, LSTM simply with more hidden sizes couldn't achieve better performance on many tasks. Though more layers LSTM presents slight improvement compared with single layer LSTM, it contains much more parameters to compute.

5 Conclusion

In this paper, a simple but effective reinforcement learning policy network architecture is proposed to introduce control theory into reinforcement learning control tasks. In general RL problems (formulated as MDP), given current state information, the three branches of our network predict next action respectively, which would be combined to compute the final action. This Proportional-Integral architecture exploits the advantage of LSTM, linear control and nonlinear control, with LSTM taking advantage of history information, linear control stabilizing system dynamics, nonlinear branch serving as global controller. Sufficient comparative and ablation experiments demonstrate the proposed model outperform existing models on various RL tasks especially continuous control tasks.

References

1. Ang, K.H., Chong, G., Li, Y.: PID control system analysis, design, and technology. *IEEE Trans. Control. Syst. Technol.* **13**(4), 559–576 (2005)
2. Bakker, B.: Reinforcement learning with long short-term memory. In: *Advances in Neural Information Processing Systems*, pp. 1475–1482 (2002)
3. Brockman, G., et al.: *OpenAI Gym* (2016)
4. Dhariwal, P., et al.: *OpenAI Baselines* (2017). <https://github.com/openai/baselines>
5. Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., Levine, S.: Composable deep reinforcement learning for robotic manipulation. *arXiv preprint arXiv:1803.06773* (2018)
6. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560* (2017)
7. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. *J. Artif. Intell. Res.* **4**, 237–285 (1996)
8. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. *CoRR abs/1412.6980* (2014). <http://arxiv.org/abs/1412.6980>
9. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* **17**(1), 1334–1373 (2016)
10. Lewis, F.L., Vrabie, D.: Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst. Mag.* **9**(3), 32–50 (2009)
11. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. *US Patent App.* 15/217,758, 26 January 2017
12. Lipton, Z.C., Berkowitz, J., Elkan, C.: A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* (2015)
13. Mahmood, A.R., Korenkevych, D., Komer, B.J., Bergstra, J.: Setting up a reinforcement learning task with a real-world robot. *arXiv preprint arXiv:1803.07067* (2018)
14. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: *International Conference on Machine Learning*, pp. 1928–1937 (2016)
15. Mnih, V., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529 (2015)
16. Plappert, M., et al.: Multi-goal reinforcement learning: challenging robotics environments and request for research (2018)
17. Rajeswaran, A., Lowrey, K., Todorov, E.V., Kakade, S.M.: Towards generalization and simplicity in continuous control. In: *Advances in Neural Information Processing Systems*, pp. 6553–6564 (2017)
18. Rummery, G.A., Niranjan, M.: *On-line Q-learning using connectionist systems*, vol. 37. University of Cambridge, Department of Engineering (1994)
19. Salih, A.L., Moghavvemi, M., Mohamed, H.A., Gaeid, K.S.: Modelling and PID controller design for a quadrotor unmanned air vehicle. In: *2010 IEEE International Conference on Automation Quality and Testing Robotics (AQTR)*, vol. 1, pp. 1–5. *IEEE* (2010)
20. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: *International Conference on Machine Learning*, pp. 1889–1897 (2015)
21. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015)

22. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) (2017)
23. Shalev-Shwartz, S., Ben-Zrihem, N., Cohen, A., Shashua, A.: Long-term planning by short-term prediction. arXiv preprint [arXiv:1602.01580](https://arxiv.org/abs/1602.01580) (2016)
24. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. *Nature* **529**(7587), 484–489 (2016)
25. Srouji, M., Zhang, J., Salakhutdinov, R.: Structured control nets for deep reinforcement learning. arXiv preprint [arXiv:1802.08311](https://arxiv.org/abs/1802.08311) (2018)
26. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: *Advances in Neural Information Processing Systems*, pp. 1057–1063 (2000)
27. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N.: Dueling network architectures for deep reinforcement learning. arXiv preprint [arXiv:1511.06581](https://arxiv.org/abs/1511.06581) (2015)
28. Watkins, C.J., Dayan, P.: Q-learning. *Mach. Learn.* **8**(3–4), 279–292 (1992)
29. Williams, J.D., Zweig, G.: End-to-end LSTM-based dialog control optimized with supervised and reinforcement learning. arXiv preprint [arXiv:1606.01269](https://arxiv.org/abs/1606.01269) (2016)
30. Wu, C., et al.: Variance reduction for policy gradient with action-dependent factorized baselines. arXiv preprint [arXiv:1803.07246](https://arxiv.org/abs/1803.07246) (2018)
31. Wu, Y., Mansimov, E., Grosse, R.B., Liao, S., Ba, J.: Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. In: *Advances in Neural Information Processing Systems*, pp. 5285–5294 (2017)