Trust Region Policy Optimization Regularized via PID Control

David S. Hippocampus*
Department of Computer Science
Cranberry-Lemon University
Pittsburgh, PA 15213
hippo@cs.cranberry-lemon.edu

Abstract

A large portion of popular RL methods can be categorized into on-policy learning, where TRPO is a classical method achieving success in wide range of applications. However, TRPO still has several problems on the computation and exploration. In order to make sure the updated policy lying in the trust region, it has to compute the inverse of Fisher information matrix, incurring large computational overhead. And since every action is drawn from the output distribution of itself, its exploration capacity is not satisfactory in some complex environments. In this work, we propose to formulate the TRPO in a KL-regularized formula and tune the Lagrangian multiplier by PID control method. To augment its exploration, we introduce a framework of multiple actors learning in parallel. The empirical experiments show the benefits of our methods over previous work.

Policy gradient is an important category of methods in reinforcement learning. It aims to search the optimal policy by following the gradient of its objective function in the policy space. It has achieved in various of popular environments, including Atari games and Mujoco Gym. Generally it can be divided into on-policy learning and off-policy learning. On-policy learning means the experience tuples used to update the policy are all from the learning policy itself, while in off-policy learning experience tuples can be sampled by different policies, which are always stored in a replay buffer. However, both on-policy and off-policy learning have their own disadvantages. Generally, on-policy learning needs very high sample complexity, since it doesn't have good exploration capacity, and every sample used to update the policy is sampled from itself. Off-policy learning is often more sample efficient, but its learning process is unstable and requires extensive tweaking of many hyper-parameters.

In this work, we propose a novel framework for on-and-off policy learning, combining on-policy and off-policy learning together. Our research is primarily based on the modifications of TRPO. There are primarily two contributions: 1) in order to avoid computing the inverse of Fisher information matrix (FIM), we formulate the KL divergence as a regularization term in the objective, and use PID control to adaptively adjust the KL divergence distance between the updated and previous policy; 2) we extend this method to multi-actor case, integrating samples from different actors (policies) in an on-and-off policy manner, which can boost the exploration of the proposed method.

1 Literature Overview

The goal of the agent in reinforcement learning (RL) is to obtain an optimal policy that can maximize the expected cumulative reward in the long term by interacting with the environment. Policy gradient methods start with a mapping from a finite-dimensional parameter space to the space of policies,

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

i.e., each policy that we consider is associated with a parameter vector θ, and thus we can denote the policy as π_θ. For the performance function $\rho(\theta) = \rho^{\pi_\theta}$, if it is differentiable with regard to the parameter vector θ and we can obtain the gradient of the performance function ρ, then one can use the the gradient ascent to update θ so as to improve the performance objective [8, 16, 19]. However, directly optimizing the policy by vanilla policy gradient methods may incur large policy changes, which can result in performance collapse due to unlimited updates.

Trust Region Policy Optimization (TRPO) [22] and Proximal Policy Optimization (PPO) [23] are two on-policy methods that optimize a surrogate function in a conservative way at each iteration. It is shown in [22] that TRPO can guarantee the policy monotonically improve at each iteration. Besides, the using of some approximations makes the TRPO a practical algorithm so that it is applicable to all general stochastic policy classes. However, the TRPO is an on-policy algorithm, which may converge prematurely to local optima in environments with high-dimensional or sparse reward. The reason lies in the fact that on-policy methods can only learn from what they collect so that they can particularly suffer from insufficient exploration ability [5]. There are some researches which aim to improve exploration, e.g., count-based exploration [7] and intrinsic motivation [4]. Yet, these strategies either introduce sensitive parameters that require careful tuning on tasks [13], or require learning additional complex structures to estimate the novelty of states.

Evolutionary algorithms (EAs) can effectively deal with the temporal credit assignment with sparse rewards, lack of effective exploration, and brittle convergence properties all together [13]. EAs use the fitness metric that consolidates the return of an entire episode [21], so that we do not encounter the temporal credit assignment problem. Based on the operations of the selection and mutation, the regions of the policy state will tend to have higher episode-wide return. Besides, the population-based approaches used in EAs help to improve the exploration ability. That is, by collecting diverse samples with a population of agents, more diverse samples can be collected so that the exploration ability is improved [25]. The robustness and stable convergence properties are also enhanced by the redundancy in a population for the EAs. However, the sample complexity of EAs is typically high and they are not efficient to solve the optimization problems with a large number of parameters, since the gradient information is not exploited [13].

Ref. [13] actually combined the evolutionary algorithm with the gradient based method, deep deterministic policy gradient (DDPG) [15]. The DDPG algorithm can provide possibly good policies for the EAs to select. The DDPG periodically inject gradient information into the EA, so that the sample complexity is decreased, making the algorithm efficient to solve the optimization problem with a large amount of parameters. On the other hand, the DDPG can employ the EA to improve the exploration ability. However, the DDPG is an off-policy algorithm, which may suffer from the stability problem. Considering the fact that the TRPO is an on-policy method that can guarantee the monotonically policy improvement [22], we intend to use the TRPO algorithm combined with EA to improve both the convergence performance and the exploration ability.

Control theory, such as non-linear control, has been applied to RL problems [9, 2, 18, 14, 24]. Specifically, in [9] authors reinterpreted first-order gradient optimization as a dynamical system. And [2] interpreted SGD as P-control and momentum methods as PI-control. They introduced a derivative term, based on the change in the gradient, and applied their resulting PID controller to improve optimization of deep convolutional networks. Generally, the literature in the intersection of RL and control is scarce.

2 Preliminaries

A Markov decision process (MDP) is defined by $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}, \mathcal{A}$ represent the set of states and actions, $p(s'|s, a)$ denotes the transition probability from state s to state s' if the action a is taken, $r(s, a)$ denotes the corresponding immediate reward received by the agent, and γ is the discount factor satisfying $\gamma \in [0, 1)$, respectively. If we consider the case that the policy can be parameterized by $theta$, then the learning aim is to find the optimal policy parameter whose corresponding policy π_θ can maximize the expected discounted return defined as $J(\pi_\theta) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t|\pi_\theta]$.

TRPO [22] is a method that searches the policy parameters θ by optimizing a surrogate function. In order to guarantee the monotonically improvement for the policy, the KL-divergence of the policies before and after an iteration is constrained by a predefined value. The basic formula is shown as

follows.

$$\max_{\theta} \mathcal{L}_{\pi_{\theta_{\text{old}}}}(\pi_\theta) = \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t^{\pi_{\theta_{\text{old}}}}(s_t, a_t)\right] \tag{1}$$

$$\text{s.t.} \quad \mathbb{E}_t\Big[D_{\text{KL}}\big(\pi_\theta(\cdot|s_t)\|\pi_{\theta_{\text{old}}}(\cdot|s_t)\big)\Big] \leq \delta, \tag{2}$$

where $\mathbb{E}_t[\ldots]$ is the average over a finite collection of experience samples, $A_t^{\pi_{\theta_{\text{old}}}}(s_t, a_t) = Q_t^{\pi_{\theta_{\text{old}}}}(s_t, a_t) - V_t^{\pi_{\theta_{\text{old}}}}(s_t)$ and $Q_t^{\pi_{\theta_{\text{old}}}}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}}\Big[V_t^{\pi_{\theta_{\text{old}}}}(s_{t+1})\Big]$ denote the state-action value function and the advantage function, respectively. For the TRPO, each policy iteration leads to an improved policy.

Similarly, PPO [23] is a method that finds the optimal policy by optimizing the surrogate function but only involves the first-order optimization using stochastic gradient descent. It is a simpler method compared with the TRPO since the TRPO typically needs to solve a second-order optimization problem. In order to ensure stable policy updates, a KL-penalized or clipped version of the objective function are subjected to maximize. For the objective function in the clipped version, the PPO algorithm seeks to maximize

$$\mathcal{L}_{\pi_{\theta_{\text{old}}}}(\pi_\theta) = \mathbb{E}_t\Big[\min(r_t(\pi_{\theta_{\text{old}}}, \pi_\theta)A_t, \text{clip}(r_t(\pi_{\theta_{\text{old}}}, \pi_\theta), 1-\epsilon, 1+\epsilon)A_t)\Big] \tag{3}$$

where $r_t(\pi_{\theta_{\text{old}}}, \pi_\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ and ϵ represent the probability ratio and the parameter for clipping, respectively.

3 PID Controlled KL-regularized On-policy Learning

In TRPO [22] computing the inverse of Fisher information matrix is the major bottleneck on improving the sampling efficiency, especially the policy is realized by deep neural networks having large amount of trainable parameters. A natural idea is to formulate the KL divergence constraint in (2) as a regularization term in the objective, multiplied by a coefficient β, as below. And the gradients of both average advantage and KL-divergence are used to update the policy. Specifically, we do not need to strictly enforce the trust region constraint, but approximately satisfies it by performing multiple steps of SGD on the objective function of the optimization problem (4).

$$\max_{\theta} \mathcal{L}_{\pi_{\theta_{\text{old}}}}(\pi_\theta) = \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t^{\pi_{\theta_{\text{old}}}}(s_t, a_t)\right] - \beta \mathbb{E}_t\Big[D_{\text{KL}}\big(\pi_\theta(\cdot|s_t)\|\pi_{\theta_{\text{old}}}(\cdot|s_t)\big)\Big] \tag{4}$$

This idea has already appeared in papers on KL-regularized RL [20, 1, 17]. However, in previous papers, the coefficient β in front of KL-divergence term is always fixed or exponentially decreasing. And the distance between the updated policy π_{θ_t} and the previous one $\pi_{\theta_{t-1}}$ can be dramatically varying during the learning process, so that the policy optimization cannot be stable or even lead to near-optimal policies.

In order to stabilize the learning process, we introduce the PID control technique to adaptively tune the parameter β, to make sure that the KL distance between the updated and previous policies is smaller or around the constraint in (2).

3.1 Optimal Control

Here we first regard the policy optimization iterations as system dynamics subject to an external influence, or control. A standard formulation for discrete-time systems with feedback control is:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= F(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \boldsymbol{y}_k &= Z(\boldsymbol{x}_k) \\ \boldsymbol{u}_k &= h(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_k) \end{aligned}$$

with state vector $\boldsymbol{x}$, dynamics function F, measurement outputs $\boldsymbol{y}$, applied control $\boldsymbol{u}$, and the subscript denoting the time step. The feedback rule h has access to past and present measurements. The goal in

optimal control is to design a control rule, h, that results in a sequence $\boldsymbol{y}_{0:T} = \{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_T\}$, scoring highly according to some metric C.

It is always easier to analyze and control systems with simpler dependence on the input, even though the dependence on the state is complex. Control-affine systems are a broad class of dynamical systems which are especially amenable to analysis [10]. Generally the dynamics there take the form

$$F(\boldsymbol{x}_k, \boldsymbol{u}_k) = f(\boldsymbol{x}_k) + g(\boldsymbol{x}_k)\boldsymbol{u}_k$$

where f and g can be non-linear and unknown a prior.

3.2 Tuning KL-Divergence Multiplier by PID Control

By interpreting the policy optimization as a dynamical system, the adaptive coefficient β can be regarded as a control input, and the KL-divergence threshold δ in (2) is a setpoint the system is maintaining. Thus, the policy optimization problem (4) can be formulated as a first-order dynamical system as below

$$\begin{aligned} \theta_{t+1} &= F(\theta_t, \beta_t) \\ y_t &= \hat{\mathbb{E}}\Big[D_{\text{KL}}\big(\pi_{\theta_t}(\cdot|s_t)\|\pi_{\theta_{t-1}}(\cdot|s_t)\big)\Big] \\ \beta_t &= h(y_0, \ldots, y_t, \delta) \end{aligned}$$

where $\hat{\mathbb{E}}$ is the empirical expectation, and h is the control rule on updating β_t to be determined later. Here the empirical KL-divergence distance is used as the system metric y_t, which is utilized by the feedback control rule h to update the multiplier. Define the first term in (4) as $J_R(\theta)$ and the second term as $J_{\text{KL}}(\theta)$. Then the dynamic function F above can be formulated as below

$$\theta_{t+1} = F(\theta_t, \beta_t) = (\theta_t + \eta \cdot \nabla J_R(\theta)) + \beta_t(\eta \cdot \nabla J_{\text{KL}(\theta)})$$

Based on the definitions above, now we can formulate the PID control rule [3] to update the multiplier β_t. The PID here has three components, i.e., proportional, integral, and derivative components. Obviously, the Lagrangian multiplier update rule for the KL constraint (2) can be written as

$$\beta_{t+1} = (\beta_t + K_I(J_{\text{KL}} - \delta))_+$$

with learning rate K_I and projection into $\beta \geq 0$. This can be regarded the integral component of the control rules on β_t. Besides, the proportional component will hasten the response to constraint violations and dampen oscillations. Different from the Lagrangian update, derivative control can act as anticipation of violations. It can both prevent KL-divergence overshoot and limit the rate of KL-divergence increases within the feasible region, making the policy optimization more adaptive to the local region and preventing the updated policy too far away from the previous one. Overall, the PID control rules for updating the multiplier β_t are as below,

$$\begin{aligned} \Delta &\longleftarrow \hat{D} - \epsilon, \\ \partial &\longleftarrow \hat{D} - \hat{D}_{\text{prev}} \\ I &\longleftarrow (I + \Delta)_+, \\ \beta &\longleftarrow (K_P\Delta + K_I I + K_D\partial)_+ \end{aligned}$$

where $\hat{D}_{\text{prev}}$ is the empirical distance in the last iteration. And here K_P, K_I, K_D are tuned empirically.

4 On-and-Off Policy Optimization Multiple Actors

As mentioned in the first part of this article, our primary goal is to propose new multi-actor algorithm to augment the exploration capability of on-policy learning method, such as TRPO. With new formulation of objective and multiplier updating rules as above, we have much wider space to design multi-actor interaction methods without considering the hard constraint on KL-divergence. Inspired by [12], we propose a novel population-guided parallel learning scheme for on-and-off policy learning specifically. We are assumed to have N identical parallel actors (policies), i.e.,

$\pi_{\theta^i}, i = 1, \ldots, N$, and every actor has its own learning objective as (4). Each actor has its own value function $V_{\phi^i}, i = 1, \ldots, N$. In each learning iteration, every actor samples K trajectories from the environment independently, denoted as $\mathcal{D}^i, i = 1, \ldots, N$.

Since the parameters of these actors are updated in a parallel way, the policies of all learners compose a population of N different policies. In order to utilize the diversity among this population of actors, the sampled trajectories from the best actor is exploited by all the other learners in each iteration, and the best actor is also updated periodically during the learning process. Specifically, at the end of M iterations, the best learner b is selected from the population based on the most recent episodic rewards of each learner.

Different from previous work on population-based learning [11, 12], both sampled trajectories and policy information of the best learner, i.e., $\mathcal{D}^b, \pi_{\theta^b}$, are integrated into the actor and value function of other learners. The trajectories in $\mathcal{D}^b$ can improve the cumulative reward performance of other learners. Minimizing the distance between π_{θ^b} and the actor of every other learner, can not only improve the performance of the population all together, but also prevent the diversity among the population from becoming too large, otherwise the trajectories of best learner can make the learning process of other learners unstable and diverge from the optimal solution.

Since every non-best learner needs to integrate trajectories $\mathcal{D}^b$, sampled by a different actor, we adopt the V-trace target [6] to mitigate the distribution differences among different actors. Therefore, for every non-best learner, the objective function of updating its policy π_{θ^i} can be formulated as

$$\arg\max_{\theta} \hat{\mathbb{E}}_{(s,a)\sim\mathcal{D}^i}\left[\nabla \log \pi_\theta \hat{A}^{\pi_\theta}\right] + \hat{\mathbb{E}}_{(s,a)\sim\mathcal{D}^k, k\neq}\left[\min(\rho, c)\nabla \log \pi_\theta \hat{A}^{\pi_\theta}\right]$$

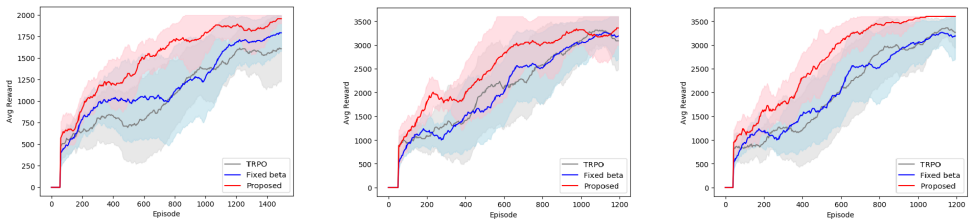
$$-\beta_t \hat{\mathbb{E}}_{(s,a)\sim\mathcal{D}^i}\left[D_{\mathrm{KL}}\left(\pi_{\theta_t^i}(\cdot|s)\|\pi_{\theta_{t-1}^i}(\cdot|s)\right)\right] - \alpha\hat{\mathbb{E}}_{(s,a)\sim\mathcal{D}^b}\left[D_{\mathrm{KL}}\left(\pi_{\theta_t^i}(\cdot|s)\|\pi_{\theta_{t-1}^b}(\cdot|s)\right)\right] \tag{5}$$

where β_t is updated with sum of two KL-divergence distances in (5). Here the empirical advantage is defined as $\hat{A}^{\pi_\theta}(s,a) = \hat{G}(s,a) - V^{\pi_\theta}(s)$, where $\hat{G}(s,a) = \sum_{k=t}^{\infty}\gamma^k r(s_k, a_k)$ starting from $(s_t, a_t) = (s, a)$, and $V^{\pi_\theta}(s)$ is the value function of actor π_θ. Besides, in order to mitigate the distribution differences, the importance ratio is defined as $\rho(s,a) = \frac{\pi_{\theta_t^i}(a|s)}{\pi_{\theta_t^b}(a|s)}$. Since the importance ratio may vary across a large magnitude, we introduce a threshold c to clip $\rho(s,a)$, to reduce the variance [6].

5 Experiments

5.1 Evaluation of TRPO Tuned by PID Control

We first conduct a set of experiments on studying the PID control on the multiplier of KL-divergence. The environment is Ant-v2, a classical game in MuJoCo Openai Gym. The action space has 8 dimensions while the observation space has 111 dimensions. We set $K_P = 100, K_I = 500, K_D = 100$ and setpoint $\delta = 0.01$. The first benchmark also has objective (4), but it sets $\beta_t = 5$, which is the optimal choice for fixed $beta_t$ from empirical experience. The second benchmark is TRPO [22], where the KL-divergence threshold $\delta = 0.01$. All the other hyper-parameters are the same for the proposed method and benchmarks.



(a) Ant (b) Hopper (c) Walker2D

Figure 1: Performance Comparison on Ant, Hopper, Walker2D

5.2 Evaluation of the Multi-actor Framework

The second set of experiments studies the performance improvement of the multi-actor method. The environments are Hopper, HalfCheetah, and Ant. The performance metric is cumulative reward at the step of $1e7$, obtained from 3 random seeds. Here we tested the method with $N = 3$ learners, and in each iteration every learner samples $K = 5$ trajectories from the environment independently. The first benchmark is the method where 3 learners are learning totally independently without interaction. Both the proposed method and first benchmark use PID control to update the KL multiplier β_t. The other two benchmarks are the classical TRPO and PPO [23] where KL threshold $\delta = 0.01$ and clipping factor $\alpha = 0.2$.

Table 1: Summary of quantitative results. All results correspond to the original exact reward defined in OpenAI Gym

	Hopper	HalfCheetah	Ant
PPO	2281.1 ± 268.1	2613.2 ± 202.8	1492.1 ± 126.9
TRPO	2156.7 ± 132.2	3211.5 ± 590.0	1918.6 ± 221.9
Parallel	2521.4 ± 180.2	2652.2 ± 322.1	1821.6 ± 173.2
Ours	$\mathbf{3021.3 \pm 172.2}$	$\mathbf{2925.7 \pm 133.1}$	$\mathbf{2537.1 \pm 158.1}$

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

[2] Wangpeng An, Haoqian Wang, Qingyun Sun, Jun Xu, Qionghai Dai, and Lei Zhang. A pid controller approach for stochastic optimization of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8522–8531, 2018.

[3] Karl Johan Åström, Tore Hägglund, and Karl J Astrom. *Advanced PID control*, volume 461. ISA-The Instrumentation, Systems, and Automation Society Research Triangle . . . , 2006.

[4] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, pages 1471–1479, 2016.

[5] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. *arXiv preprint arXiv:1802.05054*, 2018.

[6] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[7] Justin Fu, John Co-Reyes, and Sergey Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2577–2587, 2017.

[8] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[9] Bin Hu and Laurent Lessard. Control interpretations for first-order optimization methods. In *2017 American Control Conference (ACC)*, pages 3114–3119. IEEE, 2017.

[10] Alberto Isidori. *Nonlinear control systems*. Springer Science & Business Media, 2013.

[11] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

[12] Whiyoung Jung, Giseung Park, and Youngchul Sung. Population-guided parallel policy search for reinforcement learning. *arXiv preprint arXiv:2001.02907*, 2020.

[13] Shauharda Khadka and Kagan Tumer. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1188–1200, 2018.

[14] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[15] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4213–4220, 2019.

[16] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[17] Boyi Liu, Qi Cai, Zhuoran Yang, and Zhaoran Wang. Neural trust region/proximal policy optimization attains globally optimal policy. In *Advances in Neural Information Processing Systems*, pages 10565–10576, 2019.

[18] Guan-Horng Liu and Evangelos A Theodorou. Deep learning theory review: An optimal control and dynamical systems perspective. *arXiv preprint arXiv:1908.10920*, 2019.

[19] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[20] William H Montgomery and Sergey Levine. Guided policy search via approximate mirror descent. In *Advances in Neural Information Processing Systems*, pages 4008–4016, 2016.

[21] Linqiang Pan, Cheng He, Ye Tian, Handing Wang, Xingyi Zhang, and Yaochu Jin. A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 23(1):74–88, 2018.

[22] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[24] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.

[25] G Zames, NM Ajlouni, NM Ajlouni, NM Ajlouni, JH Holland, WD Hills, and DE Goldberg. Genetic algorithms in search, optimization and machine learning. *Information Technology Journal*, 3(1):301–302, 1981.