ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-4485-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICASSP48485.2024.10446137

A META-PRECONDITIONING APPROACH FOR DEEP Q-LEARNING

Spilios Evmorfos , Athina P. Petropulu

Rutgers, The State University of New Jersey
Department of Electrical and Computer Engineering

ABSTRACT

Deep Q-learning stands as an integral component within modern deep reinforcement learning algorithms. Notwithstanding its recent successes, deep Q-learning can be susceptible to instability and divergence, especially when combined with off-policy learning and bootstrapping, a combination also referred to as the "deadly triad". The current work introduces a novel learning process that aligns with the flow of gradient-based meta-learning algorithms and is designed to be performed prior to the application of deep Q-learning. The primary goal of the proposed learning process is to instill favorable generalization properties within the Q-function approximator, by conditioning its corresponding Neural Tangent Kernel. The proposed approach is applied on a sample of the environments of the DeepMind Control Suite and provides about 15% improvement in average reward accumulation.

***Index Terms*—** deep Q-learning, reinforcement learning, meta-learning, Neural Tangent Kernels

1. INTRODUCTION

The advent of deep Q-learning (DQL) stands as a pivotal factor contributing to the recent achievements in the field of Deep Reinforcement Learning (DRL) [1, 2]. In the realm of DQL methodologies, a function approximator is trained to estimate the value of each state-action pair within the relevant Markov Decision Process (MDP), under the assumption of an optimal policy. The value attributed to a specific state-action pair corresponds to the discounted sum of rewards that the agent is anticipated to collect when positioned at that pair and subsequently adhering to the optimal policy. Given that the policy is either greedy with respect to the estimated value [3], or represents a function approximator that is updated to favor the action with the highest value estimate (as seen in policy gradient methods) [4, 5, 6], the accuracy of the Q-estimator becomes a crucial aspect for successful control performance.

Despite the celebrated accomplishments of DQL across domains ranging from games (e.g., Atari) [1] to robotics [7], the robustness of DQL algorithms is marred by susceptibility to divergence and instability, impeding their seamless integration [8]. The fragility of DQL is often attributed to the coexistence of three paradigms, collectively known as the "deadly triad" [9], namely, the use of a function approximator, often a neural network; the application of off-policy learning, entailing the employment of data gathered from suboptimal policies to estimate the value of the optimal policy; and the incorporation of bootstrapping, a learning paradigm wherein the Q-function estimator is iteratively updated based on a function of its prior estimates. Recognizing these challenges, there have been proposed heuristics to mitigate the catastrophic effects of the deadly triad. Proposed remedies include the implementation of target networks [1], the adoption of n-step returns [10], and the use of ensembles of Q-networks [3].

The Neural Tangent Kernel (NTK) [11] has recently been introduced as a framework for understanding the evolution of neural networks under gradient descent. The authors in [12] study the divergent behavior of DQL under the prism of the Q-network's NTK. They adopt the assumption that if the first order approximation of the deep-Q update is a contraction in the sup norm, then DQL should be stable. Under this presupposition, they prove that a sufficient condition for stable convergence of DQL is that the diagonal elements of the Q-network's NTK exhibit substantially greater magnitudes compared to their non-diagonal counterparts.

The current work proposes a learning process to be performed before DQL and preconditions the Q-function approximator in order to enforce the sufficient condition of the corresponding NTK to hold strong throughout the main DQL training loop. The proposed method is inspired by gradient-based meta-learning approaches [13] in the sense that it employs a main objective that is used for gradient computation and a meta-objective that is a function of the previously computed gradient. The main objective of the proposed learning process is a variant of the original DQL loss and the meta-objective is a function of the Q-network NTK elements, which forces the diagonal elements to be of much larger magnitude than the non-diagonal ones. Since the meta-objective is not a function of the Q-network's output but of its gradients, the meta-updates require computing higher order derivatives with respect to the Q-network's parameters, which is computationally expensive and is not supported by the current automatic differentiation toolboxes. In order to bypass the computation of higher order derivatives, we introduce a surrogate of the initial meta-objective that replaces the

Work supported by ARO under grants W911NF2110071, W911NF232010 and NSF under grants ECCS-2033433, ECCS-2320568

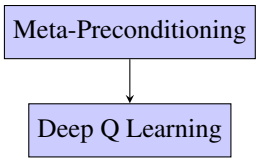

Fig. 1: Block diagram illustrating the process of meta-preconditioning before Deep Q Learning.

gradients with a zeroth-order approximation of the gradient using a Gaussian smoothing parameter. The proposed preconditioning approach is combined with the Soft Actor-Critic (SAC) [4] algorithm for continuous control that employs DQL for the critic training and is applied to a sample set of the benchmark environments of the DeepMind Control Suite (DMC) [14]. The proposed preconditioning approach provides around 15% increase in average final reward accumulation for the depicted environments. Furthermore it provides significant increase in speed of convergence and stability between seeds in comparison to the baseline implementation that employs a set of randomly initialized Q-networks.

2. DEEP Q-LEARNING

DQL is employed in cases where an agent interacts with the environment in sequences of states, actions and rewards inducing a Markov Decision Process (MDP). The MDP is defined as a tuple $\langle S, A, R, P, p, \gamma \rangle$; S denotes the state space, A the action space, $R : S \times A \to \mathbb{R}$ is the reward function, $P : S \times A \to S$ is the transition function that is dictated by the dynamics that govern the interactions of the agent with the environment, $p(\mathbf{s})$ is the distribution of the initial state, and $\gamma \in (0, 1)$ is the discount factor that implicitly measures how interested the agent is in long-term rewards.

The notion of the state-action value function is defined in conjunction to a given policy for choosing actions, π. The state-action value function Q^π of a policy π is the expected discounted sum of rewards starting from a state-action pair of the MDP and following the policy π thereafter.

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \Big[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}_t, \mathbf{a}_t) | (\mathbf{s}_0, \mathbf{a}_0) = (\mathbf{s}, \mathbf{a}) \Big] \quad (1)$$

The goal of DQL is to estimate the optimal state-action value function which is the value of each state-action pair under the optimal policy:

$$Q^*(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{s' \sim P} \Big[R(\mathbf{s}, \mathbf{a}) + \gamma \max_{a'} Q(\mathbf{s}', \mathbf{a}') \Big] \quad (2)$$

Assuming an off-policy setting with only one-step bootstrapping, which is most typically the case [1, 2, 4, 3], there exists an Experience Replay memory that contains transitions of the agent acting in the context of the MDP, $D = \big\{ \mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i \big\}_{i=1}^{N_{exp}}$. DQL involves the parameterization of the optimal state-action value function with an expressive function approximator (a deep neural network) with parameters $\mathbf{w}$, denoted as $Q_\mathbf{w}(\mathbf{s}, \mathbf{a})$). At each update step of the DQL algorithm, a batch of transitions is sampled from the Experience Replay and the parameters are updated with the following gradient descent rule:

$$\mathbf{w}' \to \mathbf{w} + \eta \mathbb{E}_{(\mathbf{s}, \mathbf{a}, \mathbf{s}', r) \sim D} \Big[\Big(Q^*_\mathbf{w}(\mathbf{s}, \mathbf{a}) - Q_\mathbf{w}(\mathbf{s}, \mathbf{a}) \Big) \nabla_\mathbf{w} Q_\mathbf{w}(\mathbf{s}, \mathbf{a}) \Big] \quad (3)$$

The parameter η is the chosen learning rate. The above gradient descent rule minimizes the Bellman loss:

$$L_{bell}(\mathbf{w}) = \mathbb{E}_{(\mathbf{s}, \mathbf{a}, \mathbf{s}', r) \sim D} \Big[\| Q_\mathbf{w}(\mathbf{s}, \mathbf{a}) - r - \gamma \max_{\mathbf{a}'} Q_\mathbf{w}(\mathbf{s}', \mathbf{a}') \|^2 \Big] \quad (4)$$

A frequently employed heuristic involves the use of another neural network, called the "target network" in order to compute the bootstrapping target of the objective. The parameters of the target network are updated with an exponential moving average of the weights of the main Q-network.

2.1. Divergence and the Q-network's NTK

In order to examine how each update step of (3) using one transition $(\mathbf{s}, \mathbf{a})$ affects the estimation of another state-action pair $(\bar{\mathbf{s}}, \bar{\mathbf{a}})$, we apply Taylor expansion of Q around $\mathbf{w}$ and keep only the first order term:

$$Q_{\mathbf{w}'}(\bar{\mathbf{s}}, \bar{\mathbf{a}}) \approx Q_\mathbf{w}(\bar{\mathbf{s}}, \bar{\mathbf{a}}) + \nabla_\mathbf{w} Q_\mathbf{w}(\bar{\mathbf{s}}, \bar{\mathbf{a}})^\mathsf{T} (\mathbf{w}' - \mathbf{w}) \quad (5)$$

By plugging (3) into (5), we obtain:

$$Q_{\mathbf{w}'}(\bar{\mathbf{s}}, \bar{\mathbf{a}}) \approx Q_\mathbf{w}(\bar{\mathbf{s}}, \bar{\mathbf{a}}) + \eta K_\mathbf{w}(\bar{\mathbf{s}}, \bar{\mathbf{a}}; \mathbf{s}, \mathbf{a}) \big(Q^*_\mathbf{w}(\mathbf{s}, \mathbf{a}) - Q_\mathbf{w}(\mathbf{s}, \mathbf{a}) \big), \quad (6)$$

where $K_\mathbf{w}(\bar{\mathbf{s}}, \bar{\mathbf{a}}; \mathbf{s}, \mathbf{a}) = \nabla_\mathbf{w} Q_\mathbf{w}(\bar{\mathbf{s}}, \bar{\mathbf{a}})^\mathsf{T} \nabla_\mathbf{w} Q_\mathbf{w}(\mathbf{s}, \mathbf{a})$ is the element of the Neural Tangent Kernel (NTK) of the Q-network [11]. The $K_\mathbf{w}$ corresponds to a symmetric matrix where:

- $K_\mathbf{w}(i, j) = K_{ij} = \nabla_\mathbf{w} Q_\mathbf{w}(\mathbf{s}_i, \mathbf{a}_i)^\mathsf{T} \nabla_\mathbf{w} Q_\mathbf{w}(\mathbf{s}_j, \mathbf{a}_j)$ $i \neq j$
- $K_\mathbf{w}(i, i) = K_{ii} = \| \nabla_\mathbf{w} Q_\mathbf{w}(\mathbf{s}_i, \mathbf{a}_i) \|^2$

At this point, we refer to the work by [12]. The authors make the following assumption. The first order approximation of the DQL update being a contraction in the sup norm is a sufficient condition for convergence. Based on the aforementioned assumption, the authors prove the following theorem:

Theorem 1 *[12] Let indices i, j refer to state-action pairs. Suppose that $K_\mathbf{w}$, ρ_i (the frequency of the i-th state-action pair in the Experience Replay), $\gamma < 1$, η satisfy the following conditions:*

$$\forall i, \quad 2\eta K_{ii} \rho_i \leq 1, \quad (7)$$

$$\forall i, \quad (1 + \gamma) \sum_{j \neq i} \| K_{ij} \| \rho_j \leq (1 - \gamma) K_{ii} \rho_i. \quad (8)$$

Then the DQL update is a contraction in the sup norm and the fixed point of the update operator is the optimal value function of the MDP.

Assuming that a sufficient amount of exploration is in place and the discount factor is close to 1, the aforementioned theorem implies that, in order to achieve stable convergence, the Q-network's NTK should have a strong diagonal and small non-diagonal elements:

$$\forall j \neq i,\ \|\nabla_{\mathbf{w}} Q_{\mathbf{w}}(\mathbf{s}_i, \mathbf{a}_i)^{\mathsf{T}} \nabla_{\mathbf{w}} Q_{\mathbf{w}}(\mathbf{s}_j, \mathbf{a}_j)\| \ll \|\nabla_{\mathbf{w}} Q_{\mathbf{w}}(\mathbf{s}_i, \mathbf{a}_i)\|^2 \tag{9}$$

3. META-PRECONDITIONING

Assuming that there is access to a collection of transitions of the agent, the goal is to develop a training regiment in order to force the diagonal elements of the NTK to be much larger than the non-diagonal elements during the process of DQL.

First, we compute the parameter vector $\mathbf{w}'$ that corresponds to the vector that is derived by the application of the DQL update rule of (3). Subsequently we assume a meta-objective, the minimization of which forces the diagonal elements of the Q-network's NTK to be of significantly larger magnitude than the non-diagonal elements:

$$L_m(\mathbf{w}) = \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}_j, \mathbf{a}_j) \sim D} \Big[\|\nabla_{\mathbf{w}} Q_{\mathbf{w}}(\mathbf{s}_i, \mathbf{a}_i)^{\mathsf{T}} \nabla_{\mathbf{w}} Q_{\mathbf{w}}(\mathbf{s}_j, \mathbf{a}_j)\| - \|\nabla_{\mathbf{w}} Q_{\mathbf{w}}(\mathbf{s}_i, \mathbf{a}_i)\|^2 \Big] \tag{10}$$

where $(\mathbf{s}_i, \mathbf{a}_i) \neq (\mathbf{s}_j, \mathbf{a}_j)$. After computing $\mathbf{w}'$ from (3), the parameter vector $\mathbf{w}$ is updated by the following gradient descent rule:

$$\mathbf{w}'' \rightarrow \mathbf{w} - \beta \nabla_{\mathbf{w}} L_m(\mathbf{w}'), \tag{11}$$

where β is the learning rate of the meta-objective. The previously described scheme finds an initial vector $\mathbf{w}$ in the parameter space of the function approximator such that when the DQL update is performed on $\mathbf{w}$ the NTK condition ((9)) to be forced strong. Since the meta-obective is a function of the gradient of the Q-network and the $\mathbf{w}'$ is also a function of the gradient of the Q-network, even if first-order meta-updates are implemented [15], there is still the need to compute third order derivatives. Such computations carry a significant computational overhead and are not typically supported by contemporary automatic differentiation frameworks [16]. In order to bypass that caveat we invoke a zeroth-order approximation of the gradient of the Q-network with respect to $\mathbf{w}$ that employs finite differences and a Gaussian smoothing parameter: In particular, let us assume the Q-network $Q_{\mathbf{w}}(s, a)$ and $\mu > 0$ a smoothing parameter. Then the μ-smooth Q-network can be defined as follows:

$$Q^{\mu}_{\mathbf{w}}(\mathbf{s}, \mathbf{a}) \coloneqq \mathbb{E}_{U \sim \mathcal{N}(0,I)} \left[Q_{\mathbf{w}+\mu U}(\mathbf{s}, \mathbf{a}) \right] \tag{12}$$

We can safely assume that, for small enough μ, the true gradient of the critic with respect to the parameters can be approximated by the zeroth order gradient of the μ-smooth critic [17]:

$$\nabla_{\mathbf{w}} Q_{\mathbf{w}}(\mathbf{s}, \mathbf{a}) \approx \mathbb{E}_{U \sim \mathcal{N}(0,I)} \Big[\frac{Q_{\mathbf{w}+\mu U}(\mathbf{s}, \mathbf{a}) - Q_{\mathbf{w}}(\mathbf{s}, \mathbf{a})}{\mu} U \Big] \tag{13}$$

By utilizing the above zeroth order approximation of the gradient, we can formulate a surrogate of the initial meta-objective that is not a function of the gradient of the Q-network and therefore it does not require the computation of third order derivatives. For a given vector U, the surrogate can be written as :

$$L^{Sur}_m(\mathbf{w}, U, \mu) = \mathbb{E}_{(\mathbf{s}, \mathbf{a}, \mathbf{s}', \mathbf{a}') \sim D} \Big[- \| \overbrace{\frac{Q_{\mathbf{w}+\mu U}(\mathbf{s}, \mathbf{a}) - Q_{\mathbf{w}}(\mathbf{s}, \mathbf{a})}{\mu} U}^{\approx \nabla_{\mathbf{w}} Q_{\mathbf{w}}(\mathbf{s}, \mathbf{a})} \|^2 + \underbrace{\| \frac{Q_{\mathbf{w}+\mu U}(\mathbf{s}', \mathbf{a}') - Q_{\mathbf{w}}(\mathbf{s}', \mathbf{a}')}{\mu} U^T U \frac{Q_{\mathbf{w}+\mu U}(\mathbf{s}, \mathbf{a}) - Q_{\mathbf{w}}(\mathbf{s}, \mathbf{a})}{\mu} \|}_{\approx \nabla_{\mathbf{w}} Q_{\mathbf{w}}(\mathbf{s}', \mathbf{a}')^T \nabla_{\mathbf{w}} Q_{\mathbf{w}}(\mathbf{s}, \mathbf{a})} \Big] \tag{14}$$

The flow of the meta-preconditioning method is provided in **Algorithm 1**. The basic idea of the approach is that a random policy is employed to sample transitions from the MDP. The transitions are stored in the Experience Replay. At every step of the random policy, a batch of transitions is sampled from the Experience Replay. The sampled batch is used to compute the parameter vector $\mathbf{w}'$ which corresponds to the update by minimizing the Bellman. Subsequently the computed vector $\mathbf{w}'$ is used to compute the meta-objective. The meta-objective is calculated based on the zeroth order approximation of the gradient and therefore requires sampling a batch of vectors (same dimension as the parameter vector of the Q-network) from a Normal distribution. One should note that in the flow of **Algorithm 1**, there is no use of any target networks. Furthermore the flow of the proposed approach resembles the structure of the gradient-based meta-learning methods [13] proposed for preconditioning neural networks for downstream tasks.

Algorithm 1 Meta-Preconditioning

Initialize Experience Replay D, $Q_{\mathbf{w}}(\mathbf{s}, \mathbf{a})$
Set μ, η, β, M
Main Body
for `all time steps` t **do**
  compute state $\mathbf{s}_t$
  sample random action $\mathbf{a}_t$
  compute next state $\mathbf{s}_{t+1}$
  compute reward r_t
  sample random next action $\mathbf{a}_{t+1}$
  Store transition $\{\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}, \mathbf{a}_{t+1}\}$ to D
  sample a batch of transitions from D $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}', \mathbf{a}')$
  $\mathbf{w}' \rightarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L_{bell}(\mathbf{w})$
  sample M vectors U_i from $\mathcal{N}(0, I)$
  compute $L_{meta}(\mathbf{w}') = \frac{1}{M} \Sigma_{i=1}^{M} L^{Sur}_m(\mathbf{w}', U_i, \mu)$
  $\mathbf{w}'' \rightarrow \mathbf{w} - \beta \nabla_{\mathbf{w}} L_{meta}(\mathbf{w}')$
end for

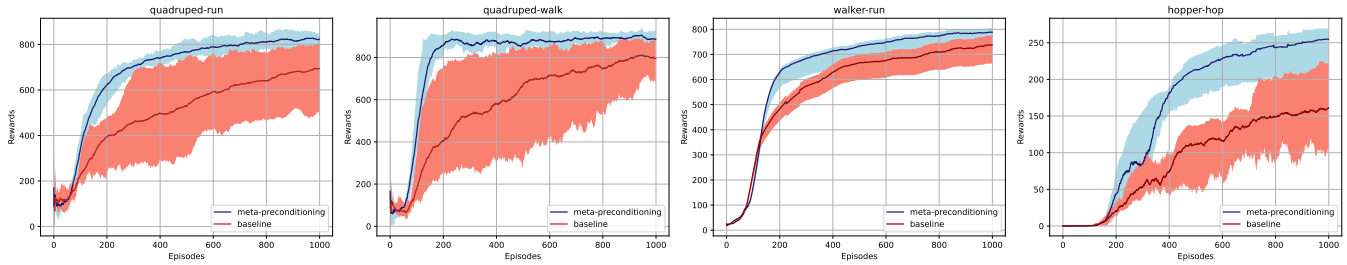

Fig. 2: Comparison of the performance of the SAC approach with a meta-preconditioned set of critics and a set of randomly initialized critics for the DMC environments. The meta-preconditioning process is not included in the plots. The environments are arranged from left to right, with the state-action dimension progressively decreasing. Each line is the average over 8 seeds.

4. EXPERIMENTS

We integrate our meta-preconditioning approach with the Soft Actor-Critic (SAC) implementation developed by [18] on the DeepMind Control Suite (DMC) environments, with proprioceptive states [14]. Specifically, our proposed meta-preconditioning method is applied to both critic networks (since the SAC implementation by [18] employs two critic neural networks) before initiating the training of the main DRL algorithm. The meta-preconditioning approach is performed for a set number of steps, approximately corresponding to the duration of 50 episodes of the main training process. The performance evaluation of the SAC algorithm, with the preconditioned critics, is benchmarked against the baseline performance of the original SAC implementation, as illustrated in Fig. 2. The learning rate β is chosen to be $1e-8$ while η is chosen to be 0.0001. We choose to sample $M = 10\ U_i s$ for estimating the gradient surrogate for the meta-objective and the μ parameter is chosen to be $1e-8$. We noticed that the vanilla gradient descent optimizer performs more stably than the popular Adam optimizer [19], so we employ it as the optimization scheme. The meta-preconditioning process is omitted from the plots depicted in Fig. 2.

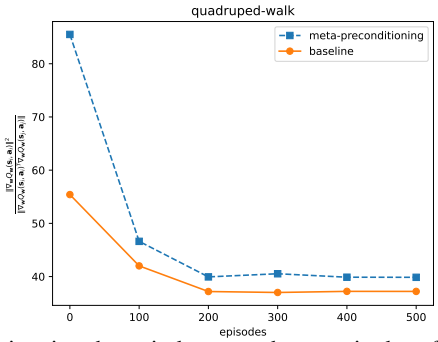

Fig. 3: Estimating the ratio between the magnitudes of the diagonal and non-diagonal elements of the Q-network's NTK both for a randomly initialized Q-network and the same Q-network that has undergone preconditioning through the proposed meta-preconditioning approach

The depicted plots demonstrate a substantial enhancement in the performance of the SAC algorithm when employing the preconditioned critics, as opposed to the baseline configuration with randomly initialized critics. The proposed approach accelerates the convergence speed while fostering a notable increase in the average reward accumulation across all depicted environments. One notable outcome of the proposed preconditioning technique is its pronounced effect on stability across different random seeds. This improvement is significant, as the overall instability observed across various seeds has posed a significant obstacle to the widespread integration of DRL algorithms in real-world engineering applications.

The visualization in Fig. 3 depicts the impact of the meta-preconditioning regiment on the gradient structure of the Q-network. The figure provides an estimation of the ratio between the magnitude of the diagonal and the off-diagonal elements of the Q-network's NTK, both with and without the proposed meta-preconditioning. The calculation of this ratio is carried out on the two critics, based on the same batch of state-action pairs for the "quadruped-walk" task of the DMC and pertains to the initial 500 training episodes of the SAC algorithm. The results presented in Fig. 3 highlight that the application of the proposed meta-preconditioning method leads to an increase in the targeted ratio. This effect is particularly prominent at the early stages of the main training loop.

5. CONCLUSIONS

Deep Q-learning is susceptible to issues such as instability and divergence. Ensuring convergence during deep Q-updates necessitates that the diagonal elements of the Q-network's NTK are notably larger than their non-diagonal counterparts. In this paper, we introduce an approach that preconditions the Q-network's NTK, fostering diagonal dominance during deep Q-update steps. To achieve this, we adopt a preconditioning strategy inspired by contemporary model-agnostic meta-learning techniques. Notably, our induced meta-objective relies on the gradients of the Q-network, rather than its output. Therefore, optimization with meta-updates introduces a significant computational overhead. In response, we propose the use of a surrogate meta-objective that leverages a zeroth-order approximation of the Q-network's gradient by finite differences and a smoothing parameter.

6. REFERENCES

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] 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.

[3] Hado Van Hasselt, Arthur Guez, and David Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2016, vol. 30.

[4] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[5] Scott Fujimoto, Herke Hoof, and David Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[6] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. Pmlr, 2014, pp. 387–395.

[7] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al., "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[8] Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine, "Implicit under-parameterization inhibits data-efficient deep reinforcement learning," *arXiv preprint arXiv:2010.14498*, 2020.

[9] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[10] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2018, vol. 32.

[11] Arthur Jacot, Franck Gabriel, and Clément Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[12] Joshua Achiam, Ethan Knight, and Pieter Abbeel, "Towards characterizing divergence in deep q-learning," *arXiv preprint arXiv:1903.08894*, 2019.

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[14] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al., "Deepmind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[15] Alex Nichol, Joshua Achiam, and John Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[16] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," 2017.

[17] Yurii Nesterov and Vladimir Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, pp. 527–566, 2017.

[18] Denis Yarats and Ilya Kostrikov, "Soft actor-critic (sac) implementation in pytorch," https://github.com/denisyarats/pytorch$_sac$, 2020.

[19] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.