# Safe and Robust Experience Sharing for Deterministic Policy Gradient Algorithms

**Baturay Saglam** [1] **Dogan C. Cicek** [1] **Furkan B. Mutlu** [1] **Suleyman S. Kozat** [1]

## Abstract

Learning in high dimensional continuous tasks is challenging, mainly when the experience replay memory is very limited. We introduce a simple yet effective experience sharing mechanism for deterministic policies in continuous action domains for the future off-policy deep reinforcement learning applications in which the allocated memory for the experience replay buffer is limited. To overcome the extrapolation error induced by learning from other agents' experiences, we facilitate our algorithm with a novel off-policy correction technique without any action probability estimates. We test the effectiveness of our method in challenging OpenAI Gym continuous control tasks and conclude that it can achieve a safe experience sharing across multiple agents and exhibits a robust performance when the replay memory is strictly limited.

## 1. Introduction

Off-policy deep reinforcement learning requires large amounts of interactions with the environment to obtain optimal policies (Schmitt et al., 2020). As the observation and action spaces of an environment start to increase and more challenging tasks are introduced, the memory requirement for the experience replay (Lin, 1992) dramatically increases (Fujimoto et al., 2019). Therefore, regardless of the experience replay sampling algorithms, with limited memory, off-policy deep RL algorithms should exhibit high-level performance for future real-world applications.

Sharing experience among concurrent agents remains an effective alternative when the available off-policy data is limited as it can allow faster convergence due to diverse exploration (Lai et al., 2020). However, learning from other agents' experiences may lead to the *extrapolation error*, a phenomenon caused by the mismatch between the distributions corresponding to the off-policy data collected by a different agent and the latest agent's policy (Fujimoto et al., 2019). The extrapolation error may lead unseen state-action pairs to be erroneously estimated and have unrealistic values (Fujimoto et al., 2019). Hence, for safe and reliable experience sharing among multiple agents, off-policy correction (or importance sampling) is required to eliminate the extrapolation error induced by other agents' experiences. Although off-policy correction and experience sharing mechanisms are well-studied artifacts for discrete (Espeholt et al., 2018; Munos et al., 2016; Schmitt et al., 2020) and continuous (Mnih et al., 2016) action domains through the action probabilities of stochastic policies, in the deterministic and continuous policy case, action probability estimation and thus, importance sampling, is not a possible option by the nature of the policies as there is not any probability distribution from which the actions are sampled.

Motivated by the possible restrictions to the allocated memory of the replay buffer and limitations of deterministic policies, we introduce an actor-critic architecture that enables a diverse and robust parallel learning. Our approach is not affected by extrapolation error by safely correcting the experiences gathered by multiple agents. An extensive set of experiments demonstrate that with only two agents that learn the environment in parallel, our architecture obtains an optimal performance when the size of the replay buffer is very limited. Moreover, we show that the introduced algorithm can significantly improve the state-of-the-art even when the replay buffer is unlimited. Ultimately, our ablation studies validate that the extrapolation occurs when the off-policy samples are not corrected, and our modifications can enable effective filtering to overcome this problem. Our code and results are available at the GitHub repository[1].

## 2. Technical Preliminaries

We follow the standard reinforcement learning paradigm, where at each discrete time step $t$, the agent observes a state $s$ and chooses an action $a$; then, it receives a reward

[1]Department of Electrical and Electronics Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey. Correspondence to: Baturay Saglam <baturay@ee.bilkent.edu.tr>.

*ICML 2022 Workshop on Responsible Decision Making in Dynamic Environments*, Baltimore, Maryland, USA, 2022. 

[1]https://github.com/baturaysaglam/DASE

$r$ and observes a new state $s'$. The policy of an agent aims to maximize the *value* defined as the expected cumulative discounted returns $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$, where $\gamma \in [0, 1)$ is a discount factor to prioritize the short-term rewards. A policy $\pi_\phi(\cdot)$, parameterized by $\phi$, is stochastic if it maps states to action probabilities, $a \sim \pi_\phi(\cdot|s)$, or deterministic if it maps states to unique actions, $a = \pi_\phi(s)$. The action-value function (Q-function or critic) evaluates the action decisions of an agent in terms of the value $R_t$. The deep Q-network, $Q_\theta$ with parameters $\theta$, estimates action-values (or Q-values).

In off-policy learning, an agent encounters transitions generated by a family of behavior policies. We consider a multiple agent case where $K$ agents explore the same environment asynchronously and store their experiences in a shared replay buffer. At every update step, the agent samples a batch of transitions through a sampling algorithm that may contain on- and off-policy samples:

$$(\boldsymbol{S}_I^{|\mathcal{B}|\times m}, \boldsymbol{A}_I^{|\mathcal{B}|\times n}, \boldsymbol{R}_I^{|\mathcal{B}|\times 1}, \boldsymbol{S}_I^{'|\mathcal{B}|\times m}) \sim \mathcal{B}_I, \quad (1)$$

$$(\boldsymbol{S}_E^{|\mathcal{B}|\times m}, \boldsymbol{A}_E^{|\mathcal{B}|\times n}, \boldsymbol{R}_E^{|\mathcal{B}|\times 1}, \boldsymbol{S}_E^{'|\mathcal{B}|\times m}) \sim \mathcal{B}_E, \quad (2)$$

where $|\mathcal{B}|$ is the number of transitions in the sampled batch, $m$ and $n$ are the state and action dimensions, respectively, and bold letters represent vectors or matrices of row vectors. We categorize transitions into the ones executed by the agent in interest and the ones executed by other agents which we call *internal (own)* and *external* experiences, respectively. We also refer to entities that corresponds to these two types of experiences as, again, *external* and *internal* entities, e.g., *external policies*, *internal actions*, *external states*. Therefore, $\mathcal{B}_I$ and $\mathcal{B}_E$ are the internal and external parts of the sampled mixed batch, respectively, yielding $\mathcal{B}_I \cup \mathcal{B}_E = \mathcal{B}$ and $\mathcal{B}_I \cap \mathcal{B}_E = \emptyset$.

## 3. Method

### 3.1. Deterministic Policy Similarity

To enable a safe experience sharing across multiple and independent agents that learn in parallel, we first aim to mitigate the extrapolation error (Fujimoto et al., 2019). We obtain this by constructing a novel policy similarity metric for deterministic policies. Before presenting the primary component of our architecture, we start with a basic assumption on the actions chosen by the deterministic policy. We assume without loss of generality that each continuous action selected by a behavioral policy is a sample of a multivariate Gaussian distribution which is not known during the training. Intuitively, each dimension in an action vector is correlated to the rest of the dimensions. This is realistic since each dimension in an action vector often has effects on the other dimensions (Todorov et al., 2012). Mainly, the mean vector represents the deterministic action chosen by the policy, and the covariance matrix represents the noise introduced by the exploration, deep function approximation, and bootstrapping in Q-learning (Watkins & Dayan, 1992).

Having our assumption made, we now show how to derive the similarity weights for the off-policy experiences. The agent samples a batch of off-policy transitions corresponding to different behavioral policies in each gradient step. We know that given the states $\boldsymbol{S}^{|\mathcal{B}|\times m}$, each action in the experience replay buffer (Lin, 1992) corresponds to a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}^{n\times 1}, \boldsymbol{\Sigma}^{n\times n})$ with mean vector $\boldsymbol{\mu}^{n\times 1}$ and covariance matrix $\boldsymbol{\Sigma}^{n\times n}$. To measure the similarity between the current policy and the policies that executed the off-policy transitions in the external batch $\mathcal{B}_E$, we first forward pass the states from $\mathcal{B}_E$ through the behavioral actor network corresponding to the current policy:

$$\hat{\boldsymbol{A}}_E^{|\mathcal{B}_E|\times n} = \pi_\phi(\boldsymbol{S}_E^{|\mathcal{B}_E|\times m}). \quad (3)$$

We now have the batch of current policy's decisions on the states from the off-policy transitions $\hat{\boldsymbol{A}}_E^{|\mathcal{B}_E|\times n}$, and the batch of past policies' decisions $\boldsymbol{A}_E^{|\mathcal{B}_E|\times n}$ from $\mathcal{B}_E$. Let $\dot{\boldsymbol{A}}_E^{|\mathcal{B}_E|\times n}$ be the batch of numerical differences in the action decisions:

$$\dot{\boldsymbol{A}}^{|\mathcal{B}_E|\times n} \coloneqq \boldsymbol{A}_E^{|\mathcal{B}_E|\times n} - \hat{\boldsymbol{A}}_E^{|\mathcal{B}_E|\times n}. \quad (4)$$

Observe that $\dot{\boldsymbol{A}}_E^{|\mathcal{B}_E|\times n}$ indicates the deviation between the current policy and previous behavioral policies of the agent that generated the off-policy transitions. To construct a multivariate Gaussian distribution from the action difference batch, let:

$$\dot{\boldsymbol{\mu}}^{n\times 1} = \frac{1}{|\mathcal{B}_E|} \sum_{i=1}^{|\mathcal{B}_E|} (\dot{\boldsymbol{A}}_i^{|\mathcal{B}_E|\times n})^\top, \quad (5)$$

$$\dot{\boldsymbol{\Sigma}}^{n\times n} = \frac{1}{|\mathcal{B}_E| - 1} \sum_{i=1}^{|\mathcal{B}_E|} \boldsymbol{a}_i^{n\times 1} (\boldsymbol{a}_i^{n\times 1})^\top, \quad (6)$$

where $\dot{\boldsymbol{A}}_i^{|\mathcal{B}_E|\times n}$ represents the action as a row vector corresponding to the $i^{\text{th}}$ transition and $\boldsymbol{a}_i^{n\times 1} = (\dot{\boldsymbol{A}}_i^{|\mathcal{B}_E|\times n})^\top - \dot{\boldsymbol{\mu}}^{n\times 1}$. Then, define the dissimilarity measure as:

$$\rho = \text{JSD}(\mathcal{N}(\dot{\boldsymbol{\mu}}^{n\times 1}, \dot{\boldsymbol{\Sigma}}^{n\times n}) \parallel \mathcal{N}(\boldsymbol{0}^{n\times 1}, \sigma \boldsymbol{I}^{n\times n})), \quad (7)$$

where JSD is the Jensen-Shannon divergence, $\sigma$ is the standard deviation of the exploration noise, and $\boldsymbol{I}$ is the identity matrix. We do not directly compare $\mathcal{N}(\dot{\boldsymbol{\mu}}^{n\times 1}, \dot{\boldsymbol{\Sigma}}^{n\times n})$ with a zero multivariate Gaussian since the policies closer to the current policy may be rejected as the actions may deviate from the policy's actual action decisions due to the additive exploration noise. Furthermore, we choose JSD for asymmetric similarity measurement as the similarity of two policy distributions should not be assumed to be directed. Although KL-divergence is well-known for penalizing a

distribution that is completely different from the distribution in interest, two policies in the same environment cannot be completely distinct (Sutton & Barto, 2018). Naturally, if all the internal and external experiences correspond to the same policy, then $\rho = 0$ and $\rho \in (0, \infty)$ otherwise. To project the similarity measure into the interval $[0, 1]$, a non-linear transformation can be applied:

$$\boldsymbol{\lambda}^{|\mathcal{B}|_E \times 1} = [e^{-\rho},\ e^{-\rho},\ \ldots,\ e^{-\rho}]^\top. \tag{8}$$

We choose the exponential function for non-linear transformation to slowly smooth the dissimiliarity. A sharp smoothing would be very greedy and may penalize the external transitions too much. Observe that two identical policies have $\boldsymbol{\lambda}^{|\mathcal{B}_E| \times 1} = \mathbf{1}$, and distinct policies have $\boldsymbol{\lambda}^{|\mathcal{B}_E \times 1} = \mathbf{0}$, making Equation (8) a similarity measure between two policies. This forms the backbone of our architecture, which we refer Deterministic Policy Similarity (DPS), summarized in Algorithm 1.

Intuitively, DPS first computes the numerical difference between the actions chosen by the current and external policies, then compares the difference with zero. This is equivalent to comparing the distributions of the current policy and the policies that executed the external transitions under the multivariate Gaussian distribution assumption. One concern with DPS may be that the minority of the transitions within the batch $\mathcal{B}_E$ may be executed by the policies very similar to the current agent's policy. Since we take the average of external action batch in computing the similarity weight, i.e., Equation (5), those transitions may be weighted by a fixed weight close to 0, which results in loss of information. Nevertheless, since the function approximators in off-policy actor-critic methods are often optimized through mini-batch learning, it should be expected that policies correspond to the majority of the transitions in $\mathcal{B}_E$ must be close to the current policy (Fujimoto et al., 2019).

### 3.2. Deterministic Actor-Critic with Shared Experience

Now, we are ready to introduce our architecture, Deterministic Actor-Critic with Shared Experience (DASE). DASE considers multiple agents, each of which explores different copies of the same environment, i.e., they learn in parallel and does not interact with each other except for a shared experience replay buffer. At every update step, each agent samples a batch of transitions and updates its actor and critic networks by combining internal gradients and DPS weighted external gradients. Through DPS, our architecture enables agents to safely use other agents' experiences by resolving the issues with stability due to the potential exploding gradients, i.e., similarity weights restricted are within the interval $[0, 1]$, and extrapolation error (Fujimoto et al., 2019), i.e., by allowing only the transitions correlated to the distribution under the current policy.

This simple architecture can accelerate learners using different GPUs and agents to be distributed across many machines for the cases in which a single learner fails to reach optimal returns, e.g., limited memory for the replay buffer. DASE can also form ensembles of deterministic actor-critic methods and sampling algorithms to solve challenging continuous control tasks by safely sharing information. However, due to the asynchronous nature of the architecture, some agents may be several updates ahead of the rest, also known as *policy lag* (Espeholt et al., 2018). Nonetheless, such a policy lag is corrected by DPS by maintaining only the external policies closer to the distribution under the current policy.

We perform extensive theoretical analysis on our approach. Notably, Theorem 3.1 provides a convergence guarantee for Q-learning (Watkins & Dayan, 1992) under DASE and Corollary 3.2 proves that DPS produces accurate importance weights such that a safe experience sharing can be achieved. All proofs are in Appendix B and the pseudocode for our hyper-parameter-free algorithm is given in Algorithm 2 through learner threads.

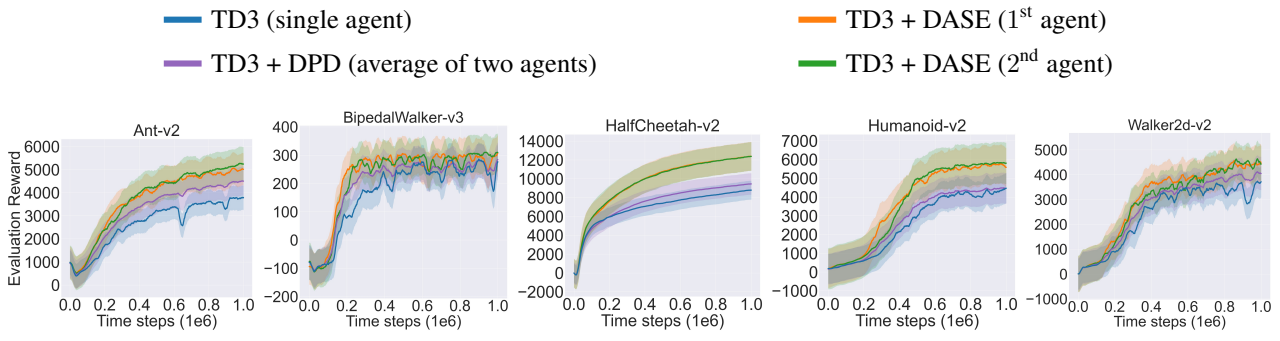
**Theorem 3.1.** *Under the Robbins-Monro stochastic convergence conditions on the learning rate $\eta$ and standard sampling requirements from the environment, Q-learning with the DASE architecture converges to the optimal value function $Q^*$.*

**Corollary 3.2.** *$\xi(s, a) \in [0, \gamma]$ is a contraction coefficient based on $(s, a)$ where $\xi(s, a) = \gamma$ if $\lambda = 0$, i.e., when there is no similarity, and close to zero when the behavioral policies corresponding to the sampled batch match the current policy.*

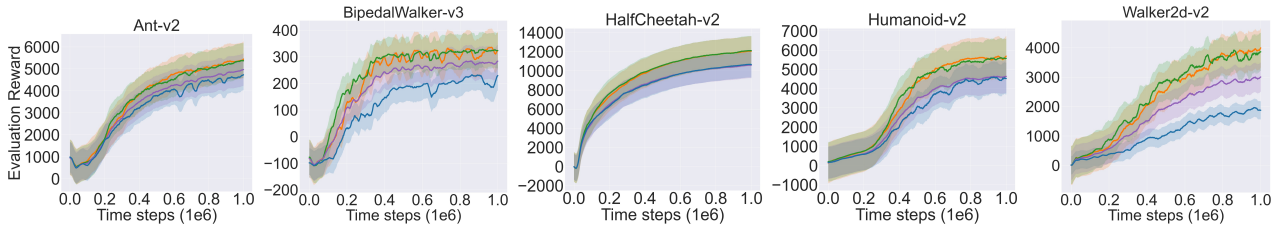
## 4. Experiments

### 4.1. Experimental Details

We conduct experiments to evaluate the effectiveness of DASE on OpenAI Gym (Brockman et al., 2016) continuous control benchmarks. We apply our method to the state-of-the-art off-policy actor-critic algorithm, Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018). Moreover, our method is compared with a single TD3 (Fujimoto et al., 2018) agent and the Dual Policy Distillation (DPD) algorithm (Lai et al., 2020), a student-student framework in which two learners operate in the same environment to investigate diverse viewpoints and extract knowledge from one another to help them learn more effectively, similar to our work. A complete list of hyper-parameters and experimental details are provided in Appendix C. In addition to TD3 (Fujimoto et al., 2018), Appendix D presents the results for DDPG (Lillicrap et al., 2016) and SAC (with deterministic actor) (Haarnoja et al., 2018), and additional continuous control tasks.




(a) **Replay Size:** 100,000




(b) **Replay Size:** 1,000,000

*Figure 1.* Learning curves for the set of OpenAI Gym continuous control tasks when replay size is 1 million and 100,000. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window smoothes curves for visual clarity.

We consider two settings of the experience replay buffer (Lin, 1992): a strictly limited (of size 100,000 transitions) and unlimited. For a fair evaluation with DPD (Lai et al., 2020) which utilizes two agents that simultaneously explore the environment, we run DASE with two agents, i.e., $K = 2$. Figure 1 depicts the experimental results under the two settings of the replay memory. Note that the curves for DPD (Lai et al., 2020) are the average of its two agents, while we depict both agents of DASE when $K = 2$ for further discussion. We discuss the computational complexity introduced by DASE in Appendix E. Moreover, a comprehensive set of ablation studies is provided to analyze the effect of each DASE component in Appendix F.

### 4.2. Discussion

From our comparative evaluations, we infer notable results. First, DASE substantially improves the TD3 algorithm (Fujimoto et al., 2018) and outperforms DPD (Lai et al., 2020) in all of the tasks tested. As expected, both agents perform similar behavior since there is no component in our algorithm that discriminates against the agents. Although a limited replay buffer does not always correspond to worse performance, a performance difference between the considered buffer settings always exists, e.g., in the BipedalWalker and Walker2d environments. This may be due to the environment dynamics, that is, some environments can be optimally learned only by the most recent collected transitions, which is explained by the fact that on-policy methods usually outperform off-policy methods in these environments (Henderson et al., 2018). Nevertheless, in such cases, our method is almost invariant to the replay buffer size, having a robust performance due to a diverse exploration and its safe experience sharing approach. Lastly, the DPD algorithm (Lai et al., 2020) exhibits a suboptimal behavior in the majority of the tasks, which we believe is caused by the decreased convergence rate due to the additional trajectory generation that introduces substantial computational overhead.

## 5. Conclusion

This paper introduces a novel continuous off-policy actor-critic architecture that employs multiple explorer agents and a shared experience replay buffer to obtain robust parallel learning when the allocated memory for the collected transitions is limited. Through a safe experience sharing among concurrent agents, it can overcome extrapolation error (Fujimoto et al., 2019) by a novel off-policy correction method. Experiments show that the introduced method can achieve state-of-the-art results while baseline algorithms fail to converge under a very limited replay memory condition. Moreover, it can also generalize to cases in which the replay buffer is unlimited, where the state-of-the-art is improved significantly. In practical applications of off-policy deep reinforcement learning where action spaces are large and continuous, we believe DASE will be an effective foothold for future approaches in attaining data efficiency and distributing agents.

## References


Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. URL https://ojs.aaai.org/index.php/AAAI/article/view/11694.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015. URL http://arxiv.org/abs/1412.6980.

Lai, K.-H., Zha, D., Li, Y., and Hu, X. Dual policy distillation. In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 3146–3152. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/435. URL https://doi.org/10.24963/ijcai.2020/435. Main track.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016. URL http://arxiv.org/abs/1509.02971.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8(3–4):293–321, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992699. URL https://doi.org/10.1007/BF00992699.

Melo, F. S. Convergence of q-learning: a simple proof, 2001. URL http://users.isr.ist.utl.pt/~mtjspaan/readingGroup/.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/mniha16.html.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/c3992e9a68c5ae12bd18488bc579b30d-Paper.pdf.

Parberry, I. *Introduction to Game Physics with Box2D*. CRC Press, Inc., USA, 1st edition, 2013. ISBN 1466565764.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-

an-imperative-style-high-performance-deep-learning-library.pdf.

Schmitt, S., Hessel, M., and Simonyan, K. Off-policy actor-critic with shared experience replay. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8545–8554. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/schmitt20a.html.

Singh, S., Jaakkola, T., Littman, M., and Szepesvári, C. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38:287–308, 03 2000. doi: 10.1023/A:1007678930559.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Watkins, C. J. C. H. and Dayan, P. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

# A. Pseudocode

**Algorithm 1** Deterministic Policy Similarity (DPS)

**Input:** $\pi_\phi, \mathcal{B}$
**Output:** $\boldsymbol{\lambda}^{|\mathcal{B}|\times 1}$
Obtain the external transitions: $(\boldsymbol{S}_E^{|\mathcal{B}|\times m}, \boldsymbol{A}_E^{|\mathcal{B}|\times n}, \boldsymbol{R}_E^{|\mathcal{B}|\times 1}, \boldsymbol{S}_E^{'|\mathcal{B}|\times m}) \sim \mathcal{B}_E$
Compute the current action decisions: $\hat{\boldsymbol{A}}_E^{|\mathcal{B}_E|\times n} = \pi_\phi(\boldsymbol{S}_E^{|\mathcal{B}_E|\times m})$
Obtain the action difference batch: $\dot{\boldsymbol{A}}^{|\mathcal{B}_E|\times n} \coloneqq \boldsymbol{A}_E^{|\mathcal{B}_E|\times n} - \hat{\boldsymbol{A}}_E^{|\mathcal{B}_E|\times n}$
Compute the mean of the the multivariate Gaussian: $\dot{\boldsymbol{\mu}}^{n\times 1} = \frac{1}{|\mathcal{B}_E|}\sum_{i=1}^{|\mathcal{B}_E|}(\dot{\boldsymbol{A}}_i^{|\mathcal{B}_E|\times n})^\top$
Compute the covariance matrix of the multivariate Gaussian: $\dot{\boldsymbol{\Sigma}}^{n\times n} = \frac{1}{|\mathcal{B}_E|-1}\sum_{i=1}^{|\mathcal{B}_E|}\boldsymbol{a}_i^{n\times 1}(\boldsymbol{a}_i^{n\times 1})^\top$
Compute the dissimilarity metric: $\rho = \mathrm{JSD}(\mathcal{N}(\dot{\boldsymbol{\mu}}^{n\times 1}, \dot{\boldsymbol{\Sigma}}^{n\times n}) \parallel \mathcal{N}(\mathbf{0}^{n\times 1}, \sigma\boldsymbol{I}^{n\times n}))$
Convert the dissimilarity to the similarity to construct the DPS weights: $\boldsymbol{\lambda}^{|\mathcal{B}|_E\times 1} = [e^{-\rho},\ e^{-\rho},\ \ldots,\ e^{-\rho}]^\top$
**return** $\boldsymbol{\lambda}^{|\mathcal{B}|\times 1}$

**Algorithm 2** Deterministic Actor-Critic with Shared Experience (DASE)

Initialize $K$ agents with actor $\pi_{\phi_i}$ and critic $Q_{\theta_i}$ networks with parameters $\phi_i$ and $\theta_i$ for $i = 1, \ldots, K$
Initialize target networks if required
Initialize global experience replay buffer $\mathcal{R}$
**for** each learner thread $i = 1, \ldots, K$ **do**
  **for** each exploration time step **do**
    Obtain transition tuple $\tau$
    Store transition tuple $\tau$ in $\mathcal{R}$
  **end for**
  **for** each training iteration **do**
    Sample a batch of transitions $\mathcal{B}$ from $\mathcal{R}$
    Obtain the DPS weights: $\boldsymbol{\lambda}^{|\mathcal{B}|\times 1}$ = **DPS**($\pi_{\phi_i}$, $\mathcal{B}$)
    Weigh the external transitions by $\boldsymbol{\lambda}^{|\mathcal{B}|\times 1}$
    Update $\phi_i$ and $\theta_i$ by both internal and weighted external transitions
    Update target networks if required
  **end for**
**end for**

# B. Missing Proofs

## B.1. Convergence Guarantee

**Lemma B.1.** *Consider a stochastic process ($\xi_t$, $\Delta_t$, $F_t$), $t \geq 0$ where $\xi_t, \delta_t, F_t : X \to \mathbb{R}$, satisfies the equations:*

$$\Delta_{t+1}(x_t) = (1 - \xi_t(x_t))\Delta_t(x_t) + \xi_t(x_t)F_t(x_t); \quad x_t \in X, t = 0, 1, 2, \ldots. \tag{9}$$

*Let $\mathcal{P}_t$ be a sequence of increasing $\sigma$-fields such that $\eta_0$ and $\Delta_0$ are $P_0$-measurable and $\xi_t$, $\Delta_t$ and $F_{t-1}$ are $\mathcal{P}_t$-measurable. For $t = 1, 2, \ldots$, assume that the following conditions hold:*

1. *The set $X$ is finite.*
2. *$\xi_t(x_t) \in [0, 1]$, $\sum_t \xi_t(x_t) = \infty$, $\sum_t(\xi_t)^2 < \infty$ with probability 1 and $\forall x \neq x_t : \xi(x) = 0$.*
3. *$||\mathbb{E}[F_t|\mathcal{P}_t]|| \leq \kappa||\Delta_t|| + c_t$ where $\kappa \in [0, 1)$ and $c_t$ converges to 0 with probability 1.*

4. $\mathrm{Var}(F_t(x_t)|\mathcal{P}_t) \leq K(1+\kappa||\Delta_t||)^2$ *where* $K$ *is a constant.*

*Where* $||\cdot||$ *denotes the maximum norm. Then,* $\Delta_t$ *converges to 0 with probability 1.*

*Proof.* See (Watkins & Dayan, 1992; Singh et al., 2000; Melo, 2001). □

**Lemma B.2.** *Given a finite MDP* $(\mathcal{S}, \mathcal{A}, p, r)$ *and the transition tuple* $(s_t, a_t, r_t, s_{t+1})$ *at time step* $t$*, the Q-learning algorithm given by the update rule:*

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \eta_t[r_t + \gamma \max_{a' \in \mathcal{A}} Q_t(s_{t+1}, a') - Q_t(s_t, a_t)], \tag{10}$$

*converges to the optimal Q-function denoted by* $Q^*$ *with probability 1 if*

$$\sum_t \eta_t = \infty, \quad \sum_t \eta_t^2 < \infty; \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{11}$$

*Proof.* The proof largely relies on Lemma B.1 (Singh et al., 2000). First, Condition 1 in Lemma B.1 is satisfied by the finite MDP by setting $X = \mathcal{S} \times \mathcal{A}$. The assumption of Robbins-Monro stochastic convergence conditions on the learning rate $\eta_t$ satisfies Condition 2 by setting $\xi_t = \eta_t$. Then, let:

$$F_t(s_t, a_t) = r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a') - Q^*(s_t, a_t), \tag{12}$$

$$\Delta_t = Q_t(s_t, a_t) - Q^*(s_t, a_t), \tag{13}$$

$$\mathcal{P}_t = \{Q_0, s_0, a_0, \eta_0, r_1, s_1, \ldots, s_t, a_t\}. \tag{14}$$

If state-action visitation and updates are performed infinitely often, and $\gamma < 1$, then by (12), (13) and (14), Condition 3 is satisfied by the contraction of the Bellman Operator $\mathcal{T}$ (Melo, 2001). Finally, Condition 4 follows from a bounded and deterministic reward function $r(s_t, a_t)$ (Melo, 2001), i.e., $r_t = r(s_t, a_t)$. Then, by Lemma B.1, as $\Delta_t$ converges to 0 with probability 1, $Q_t$ converges to $Q^*$ with probability 1.

□

**Theorem B.3.** *Under the Robbins-Monro stochastic convergence conditions on the learning rate* $\eta$ *and standard sampling requirements from the environment, Q-learning with the DASE architecture converges to the optimal value function* $Q^*$*.*

*Proof.* Follows from the proof of Lemma B.2. If the sequences of increasing $\sigma$-fields are split into the fields correponding to the internal and external sequences, convergence of Q-learning with internal transitions are already given by Lemma B.2. For external transitions, Conditions 3 and 4 are altered due to DPS weights $\lambda$, by (12), (13) and (14), we have:

$$\lambda||\mathbb{E}[F_t|\mathcal{P}_t]|| \leq \kappa\lambda||\Delta_t|| + c_t, \tag{15}$$

$$\lambda^2 \mathrm{Var}(F_t(x_t)|\mathcal{P}_t) \leq K(1+\kappa\lambda||\Delta_t||)^2. \tag{16}$$

As $\lambda \in [0,1]$, clearly (15) is satisfied. Moreover, since $\lambda^2\mathrm{Var}(F_t(x_t)|\mathcal{P}_t) \leq \lambda^2 K(1+\kappa||\Delta_t||)^2$ and $\lambda^2 K(1+\kappa||\Delta_t||)^2 \leq K(1+\kappa\lambda||\Delta_t||)^2$, we also have (16) satisfied. Furthermore, the internal and external experiences are the samples of the same finite MDP $X$, and Robbins-Monro stochastic convergence conditions also apply on the external sequences which yield Conditions 1 and 2 to be satisfied. Therefore, Q-learning with the DASE architecture converges to the optimal Q-function under the requirements of infinitely many state-action visitation and updates, and $\gamma < 1$. □

### B.2. Safe Experience Sharing

**Definition B.4.** The general expectation operator for one-step importance sampling in return-based off-policy algorithms is defined by:

$$\mathcal{H}Q(s,a) \coloneqq Q(s,a) + \mathbb{E}_\eta[r + \gamma \mathbb{E}_\pi Q(s',\cdot) - Q(s,a)], \tag{17}$$

for some non-negative one-step importance sampling coefficient $\lambda$, and any behavioral policy $\eta$, where we write $\mathbb{E}_\pi Q(s,\cdot) \coloneqq \sum_a \pi(a|s)Q(s,a)$.

**Lemma B.5.** *The difference between $\mathcal{H}Q$ and its fixed point $Q^\pi$ is expressed by:*

$$\mathcal{H}Q(s,a) - Q^\pi(s,a) = \mathbb{E}_\eta[\gamma(\mathbb{E}_\pi[(Q-Q^\pi)(s,\cdot)] - \lambda(Q-Q^\pi)(s,a))] \tag{18}$$

*Proof.* Follows from the proof of Lemma 1 in (Munos et al., 2016). First, let $\Delta Q \coloneqq Q - Q^\pi$. Then, by rewriting Eq. (17):

$$\mathcal{H}Q(s,a) = \mathbb{E}_\eta[r + \gamma(\mathbb{E}_\pi Q(s',\cdot) - \lambda' Q(s',a'))], \tag{19}$$

where $\lambda'$ is the coefficient of the next transition. As $Q^\pi$ is the fixed point of $\mathcal{H}$, we have:

$$Q^\pi(s,a) = \mathcal{H}Q^\pi(s,a) = \mathbb{E}_\eta[r + \gamma(\mathbb{E}_\pi Q^\pi(s',\cdot) - \lambda' Q^\pi(s',a'))], \tag{20}$$

from which we infer that:

$$\begin{aligned} \mathcal{H}Q(s,a) - Q^\pi(s,a) &= \mathbb{E}_\eta[\gamma(\mathbb{E}_\pi \Delta Q(s',\cdot) - \lambda' \Delta Q(s',a'))], \\ &= \gamma \mathbb{E}_\eta[\mathbb{E}_\pi \Delta Q(s,\cdot) - \lambda \Delta Q(s,a)], \\ &= \mathbb{E}_\eta[\gamma\,(\mathbb{E}_\pi \Delta Q(s,\cdot) - \lambda \Delta Q(s,a))]. \end{aligned} \tag{21}$$

□

**Theorem B.6.** *The operator $\mathcal{H}$ defined by Definition B.4 has a unique fixed point $Q^\pi$. Moreover, if for each action selected by the policy $a \in \mathcal{A}$ and sampled batch of transitions $\mathcal{B}$, we have $\lambda = \lambda(a,\mathcal{B}) \in [0, e^{-\rho}]$. Then for any Q-function $Q$, we have:*

$$||\mathcal{H}Q - Q^\pi|| \leq \gamma ||Q - Q^\pi||, \tag{22}$$

*under the current policy $\pi$.*

*Proof.* Follows from the adaptation of proof of Theorem 1 in (Munos et al., 2016) to one-step importance sampling. It is trivial to observe from Definition B.4 that $Q^\pi$ is the fixed point of the operator $\mathcal{H}$ since:

$$\mathbb{E}_{s'\sim P(\cdot|s,a)}[r + \gamma \mathbb{E}_\pi Q^\pi(s',\cdot) - Q^\pi(s,a)] = (\mathcal{T}^\pi Q^\pi - Q^\pi)(s,a) = 0, \tag{23}$$

as $Q^\pi$ is the fixed point of $\mathcal{T}^\pi$. Let $\Delta Q \coloneqq Q - Q^\pi$, and from Lemma B.5, we have:

$$\begin{aligned} \mathcal{H}Q(s,a) - Q^\pi(s,a) &= \mathbb{E}_\eta[\gamma\,(\mathbb{E}_\pi \Delta Q(s,\cdot) - \lambda \Delta Q(s,a))], && (24)\\ &= \gamma \mathbb{E}_\eta[\mathbb{E}_\pi \Delta Q(s,\cdot) - \lambda \Delta Q(s,a)], && (25)\\ &= \gamma \mathbb{E}_\eta[\mathbb{E}_\pi \Delta Q(s,\cdot) - \mathbb{E}_a[\lambda(a,\mathcal{B})\Delta Q(s,a)|\mathcal{B}]], && (26)\\ &= \gamma \mathbb{E}_\eta[\sum_b (\pi(b|s) - \eta(b|s)\lambda(b|\mathcal{B}))\Delta Q(s,b)]. && (27) \end{aligned}$$

Now, since $\lambda \in [0,1]$, we have:

$$\mathcal{H}Q(s,a) - Q^\pi(s,a) = \sum_{y,b} w_{y,b} \Delta Q(y,b), \tag{28}$$

which is a linear combination of $\Delta Q(y,b)$ weighted by:

$$w_{y,b} \coloneqq \gamma \mathbb{E}_\eta[(\pi(b|s) - \eta(b|s)\lambda(b|\mathcal{B}))\mathbb{I}\{s=y\}], \tag{29}$$

where $\mathbb{I}(\cdot)$ is the indicator function. The sum of those coefficients over $y$ and $b$ is:

$$\sum_{y,b} \omega_{y,b} = \gamma \mathbb{E}_\eta [\sum_b (\pi(b|s) - \eta(b|s)\lambda(b, \mathcal{B}))], \tag{30}$$

$$= \gamma \mathbb{E}_\eta [\mathbb{E}_a [1 - \lambda(a, \mathcal{B})|\mathcal{B}]] \tag{31}$$

$$= \gamma \mathbb{E}_\eta [1 - \lambda] \tag{32}$$

$$= \gamma - \gamma\Lambda, \tag{33}$$

where $\Lambda = \mathbb{E}_\eta[\lambda]$. As $0 \leq \Lambda \leq 1$, we have $\sum_{y,b} \omega_{y,b} \leq \gamma$. Therefore, $\mathcal{H}Q(s,a) - Q^\pi(s,a)$ is a sub-convex combination of $\Delta Q(y,b)$ weighted by non-negative coefficients $\omega_{y,b}$ which sum to at most $\gamma$. Hence, $\mathcal{H}$ is a $\gamma$-contraction mapping around $Q^\pi$. □

**Corollary B.7.** *In the proof of Theorem B.6, notice that the term $\gamma\mathbb{E}_\eta[\lambda]$ depends on $(s,a)$. Let:*

$$\xi(s,a) \coloneqq \gamma - \gamma \mathbb{E}_\eta[\lambda]. \tag{34}$$

*Then, we have:*

$$|\mathcal{H}Q(s,a) - Q^\pi(s,a)| \leq \xi(s,a)||Q - Q^\pi||. \tag{35}$$

*Thus, $\xi(s,a) \in [0,\gamma]$ is a contraction coefficient based on $(s,a)$ where $\xi(s,a) = \gamma$ if $\lambda = 0$, i.e., when there is no similarity, and close to zero when the behavioral policies corresponding to the sampled batch match the current policy.*

## C. Experimental Details

All networks are trained with PyTorch (version 1.8.1) (Paszke et al., 2019), using default values for all unmentioned hyper-parameters.

### C.1. Environment

Performances of all methods are evaluated in MuJoCo (mujoco-py version 1.50) (Todorov et al., 2012), and Box2D (version 2.3.10) (Parberry, 2013) physics engines interfaced by OpenAI Gym (version 0.17.3) (Brockman et al., 2016), using v3 environment for BipedalWalker and v2 for rest of the environments. The environment dynamics, state and action spaces, and reward functions are not pre-processed and modified for easy reproducibility and fair evaluation procedure with the baseline algorithms. Each environment episode runs for a maximum of 1000 steps until a terminal condition is encountered. The multi-dimensional action space for all environments is within the range (-1, 1) except for Humanoid, which uses the range of (-0.4, 0.4).

### C.2. Experimental Setup

All experiments are run for 1 million time steps with evaluations every 1000 time steps, where an evaluation of an agent records the average reward over 10 episodes without exploration noise and updates. We report the average evaluation return of 10 random seeds for each environment, including the initialization of behavioral policies, simulators, and network parameters. All agents are initialized with different seeds in the DASE architecture to obtain randomness in the explored state-action spaces. Unless stated otherwise, each agent is trained by one training iteration after each time step. Agents are trained by transition tuples $(s,a,r,s')$ uniformly sampled from the shared experience replay (Lin, 1992).

### C.3. Implementation

Our implementation of the off-policy actor-critic algorithms, DDPG (Lillicrap et al., 2016), SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018), and the baseline algorithm, DPD (Lai et al., 2020), closely follows the set of hyper-parameters given in the respective papers. For the implementation of TD3 (Fujimoto et al., 2018), we use the author's GitHub repository [2] for the fine-tuned version of the algorithm and the DDPG (Lillicrap et al., 2016) implementation. For the implementation of the DPD algorithm (Lai et al., 2020), we use the author's GitHub repository [3]. We also give the hyper-parameter setting given in (Fujimoto et al., 2018) for the sake of comparison in Table 2. The SAC algorithm (Haarnoja et al., 2018) follows

[2] https://github.com/sfujim/TD3

[3] https://github.com/kiminh/dual-policy-distillation

the same setup and hyper-parameter settings for the deterministic policy, as given in the paper. We implement the DASE architecture on top of the baseline algorithms separately. The implementation distributes the agents to different threads while each agent can access the shared experience replay (Lin, 1992) contained in the RAM.

### C.4. Architecture and Hyper-parameter Setting

Different from the paper, we increase the batch size in DDPG algorithm (Lillicrap et al., 2016) to 256 in order for agents to sufficiently see other agents' experiences and replace Ornstein–Uhlenbeck exploration noise with a zero-mean Gaussian with a standard deviation of 0.1. SAC (Haarnoja et al., 2018) follows the same hyper-parameter setting given in the paper except for the deterministic policy. As no exploration noise is applied to the stochastic actor of SAC (Haarnoja et al., 2018), we add Gaussian noise with a standard deviation of 0.1 to the actions selected by the deterministic policy of SAC (Haarnoja et al., 2018). For DPD (Lai et al., 2020), we apply the same set of hyper-parameters in DDPG + DPD (Lai et al., 2020) to TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018). Shared and algorithm-specific hyper-parameters are given in Table 1 and 2, respectively.

### C.5. Hyper-parameter Optimization

No hyper-parameter optimization was performed on DDPG (Lillicrap et al., 2016) and TD3 (Fujimoto et al., 2018). For SAC (Haarnoja et al., 2018), reward scale for the LunarLanderContinuous and BipedalWalker environments as they are not presented in the original paper. We tried $5$, $10$, and $20$ for the reward scale. The reward scale value of $5$ is the one that gave the highest average return over the last 10 evaluations over 10 trials of 1 million time steps, for both environments.

### C.6. Evaluation

Evaluations occur every 1000 steps, where an evaluation is an average reward over 10 episodes without exploration noise and network updates. We utilize a new environment with a fixed seed (the training seed + a constant) for each evaluation to decrease the variation caused by different seeds. Hence, each evaluation uses the same set of initial start states.

### C.7. Visualization

Learning curves are used to show performance, and they are given as an average of 10 trials with a shaded zone added to reflect a half standard deviation across the trials. The curves are smoothed uniformly over a sliding window of 25 evaluations for visual clarity.

*Table 1.* Shared hyper-parameters.

| HYPER-PARAMETER | VALUE |
|---|---|
| ACTOR REGULARIZATION | NONE |
| OPTIMIZER | ADAM (KINGMA & BA, 2015) |
| NONLINEARITY | RELU |
| DISCOUNT FACTOR ($\gamma$) | 0.99 |
| GRADIENT CLIPPING | FALSE |
| NUMBER OF HIDDEN LAYERS (ALL NETWORKS) | 2 |

*Table 2.* Algorithm specific hyper-parameters used for the implementation of the baseline algorithms.

| HYPER-PARAMETER | DDPG | SAC | TD3 |
|---|---|---|---|
| CRITIC LEARNING RATE | $10^{-3}$ | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ |
| CRITIC REGULARIZATION | $10^{-2} \times \|\|\theta\|\|^2$ | NONE | NONE |
| ACTOR LEARNING RATE | $10^{-4}$ | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ |
| TARGET UPDATE RATE ($\tau$) | $10^{-3}$ | $5 \times 10^{-3}$ | $5 \times 10^{-3}$ |
| BATCH SIZE | 256 | 256 | 256 |
| UPDATES PER OPTIMIZATION STEP | 1 | 1 | 1 |
| CRITIC UPDATE INTERVAL | 1 | 1 | 1 |
| ACTOR UPDATE INTERVAL | 1 | 1 | 1 |
| REWARD SCALING | 1 | 5 (20 FOR HUMANOID) | 1 |
| NORMALIZED OBSERVATIONS | TRUE | FALSE | FALSE |
| EXPLORATION POLICY | $\mathcal{N}(0, 0.1)$ | $\mathcal{N}(0, 0.1)$ | $\mathcal{N}(0, 0.1)$ |
| START (EXPLORATION) TIME STEPS | 25000 | 25000 | 25000 |
| NUMBER OF HIDDEN UNITS IN THE FIRST LAYER | 400 | 256 | 256 |
| NUMBER OF HIDDEN UNITS IN THE SECOND LAYER | 300 | 256 | 256 |

## D. Complete Evaluation Results

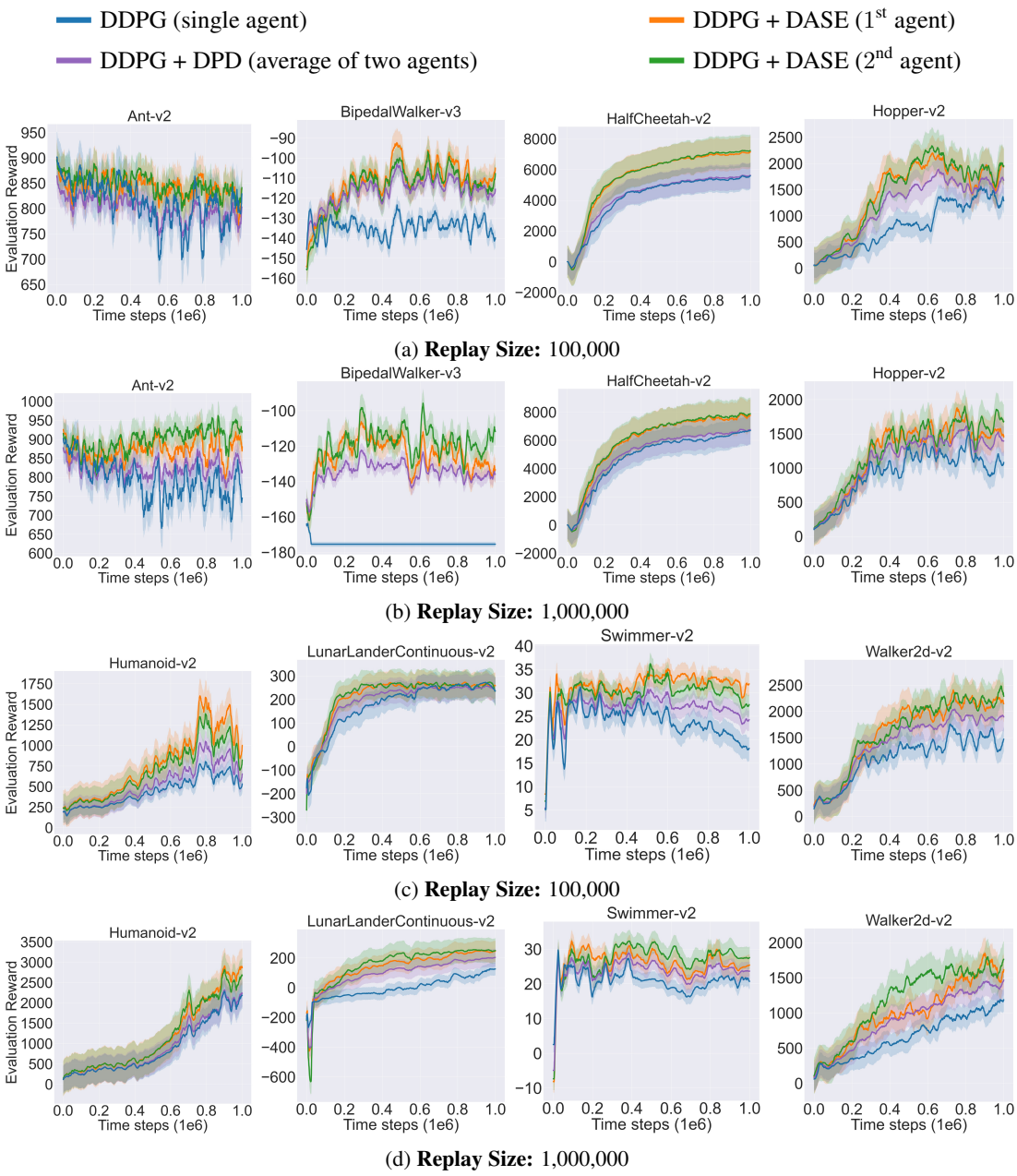
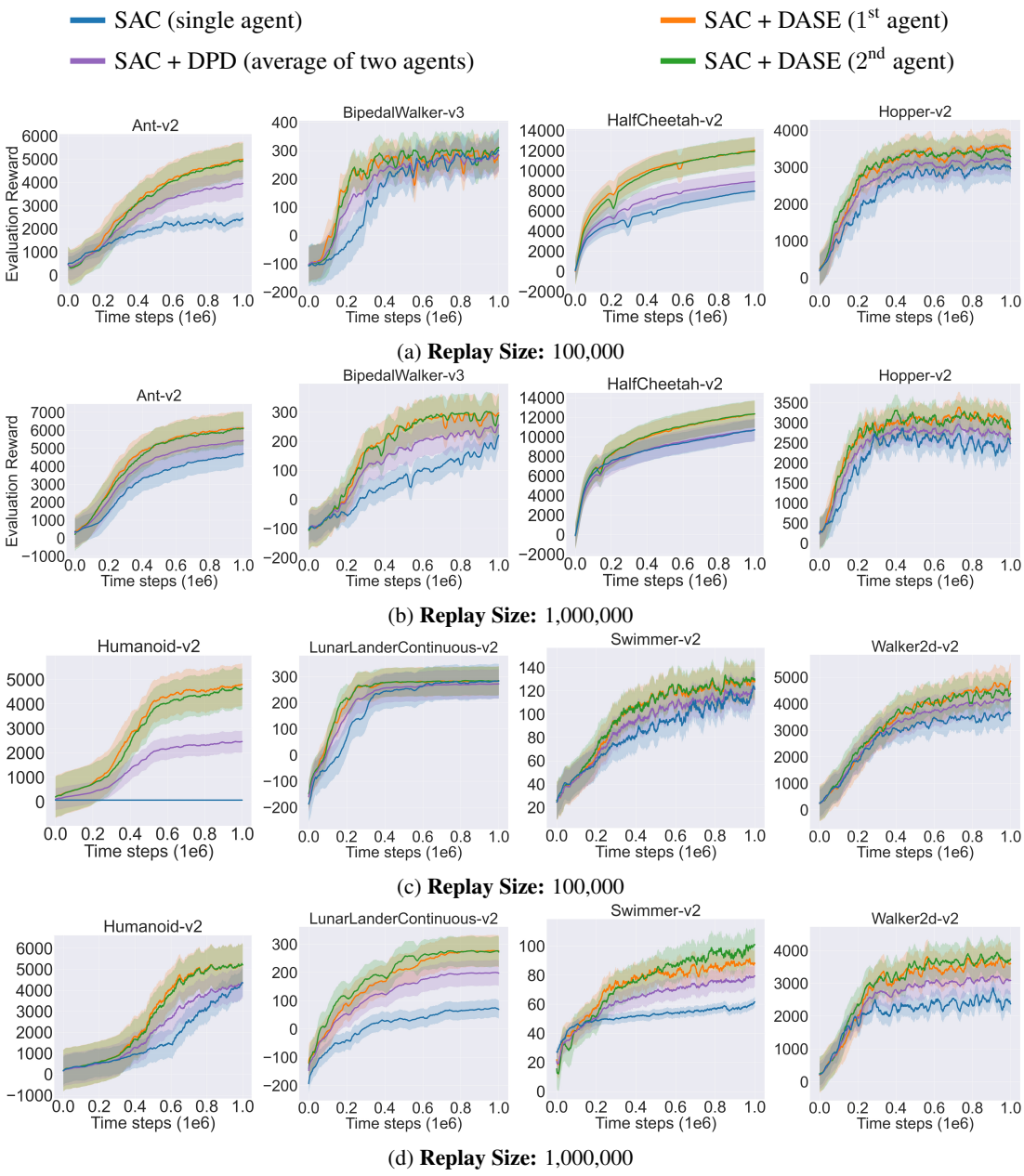



*Figure 2.* Learning curves for the set of OpenAI Gym continuous control tasks under the DDPG algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window smoothes curves for visual clarity.



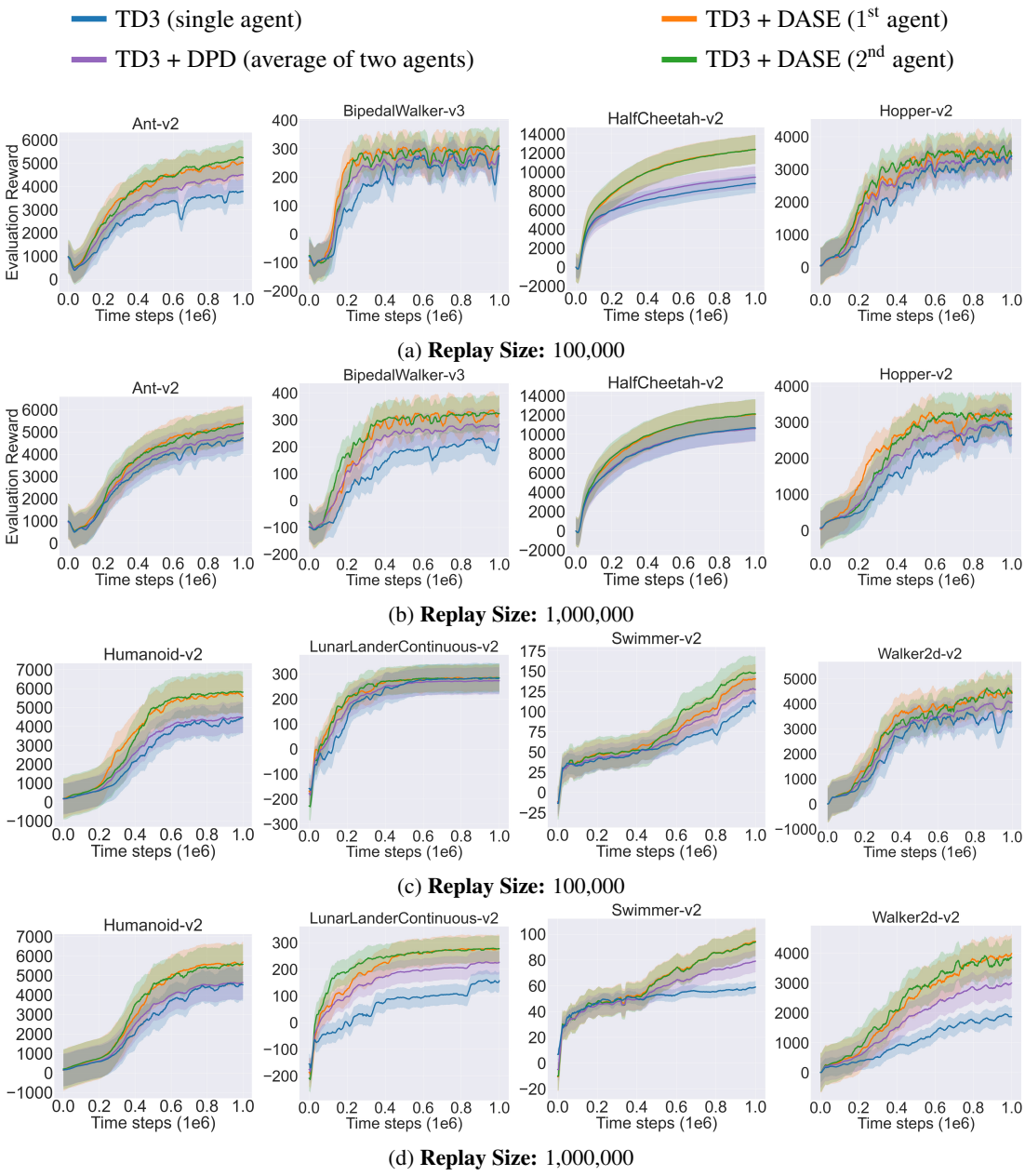

(a) **Replay Size:** 100,000

(b) **Replay Size:** 1,000,000

(c) **Replay Size:** 100,000

(d) **Replay Size:** 1,000,000

*Figure 3.* Learning curves for the set of OpenAI Gym continuous control tasks under the SAC algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window smoothes curves for visual clarity.




(a) **Replay Size:** 100,000

(b) **Replay Size:** 1,000,000

(c) **Replay Size:** 100,000

(d) **Replay Size:** 1,000,000

*Figure 4.* Learning curves for the set of OpenAI Gym continuous control tasks under the TD3 algorithm. The shaded region represents half a standard deviation of the average evaluation return over 10 random seeds. A sliding window smoothes curves for visual clarity.

## E. Computational Complexity Results

Table 3 contains the average memory required and time to run baseline algorithms and DASE overall environments, seeds, and the two settings of experience replay buffer (Lin, 1992). We report the computational complexity results when the DASE agents are distributed to multi-threads and multi-processes.

If multi-threads are used, the memory allocation for RAM and GPU does not change as the agents are trained with the same memory allocation for a single agent. For multi-processes, the memory allocation increases for both RAM and GPU. We find that GPU memory allocation is doubled, and RAM allocation is slightly less than the double of the RAM requirement for a single agent due to the shared experience replay (Lin, 1992). We expect the run time of DASE with threads to be approximately the quadruple of the baseline run time and for processes to be double. However, the run time increase for both distribution types is greater than the initial expectations. Such a significant increase is due to the simulation benchmarks we use to conduct our experiments. These simulations run on the CPU. Therefore, latency due to the data marshaling increases the run time and becomes a dominant factor when multiple agents are employed to explore the same environment.

Nonetheless, our algorithm can attain optimal evaluation returns while baseline algorithms may suffer from divergence or be stuck at suboptimal policies. Hence, there is a trade-off between our architecture's computational complexity and performance. Such a trade-off should be carefully handled by considering the environment complexity, such as determining the number of agents by considering how challenging the environment is to be explored and solved or to use threads or processes.

*Table 3.* Average memory requirement and run time for the baseline algorithms with and without DASE when agents are distributed to multi-threads and multi-processes. Run time is computed over 1 million time steps. Required memories are in megabytes and running times are in minutes.

| ALGORITHM | MEMORY (RAM) | MEMORY (GPU) | RUN TIME | RUN TIME INCREASE (%) |
|---|---|---|---|---|
| DDPG | 4056 | 1271 | 81.75 | - |
| SAC | 4087 | 1281 | 99.20 | - |
| TD3 | 4012 | 1273 | 95.67 | - |
| DDPG + DASE (THREAD) | 4102 | 1283 | 401.38 | 491% |
| SAC + DASE (THREAD) | 4118 | 1283 | 497.25 | 501% |
| TD3 + DASE (THREAD) | 4134 | 1287 | 465.41 | 486% |
| DDPG + DASE (PROCESS) | 7862 | 2542 | 216.98 | 265% |
| SAC + DASE (PROCESS) | 7924 | 2562 | 269.55 | 271% |
| TD3 + DASE (PROCESS) | 7774 | 2546 | 254.48 | 266% |

## F. Ablation Studies

*Table 4.* Ablation results when the replay size is 100,000. Bold values represent the maximum for each environment. Scores represent the average return over all agents of DASE.

| METHOD | ANT | HALFCHEETAH | LUNARLANDERCONTINUOUS | SWIMMER |
|---|---|---|---|---|
| DASE ($K = 2$) | 5123.03 | 12375.37 | 283.95 | 144.28 |
| DASE ($K = 5$) | 5490.69 | 12764.66 | 306.43 | 155.38 |
| DASE ($K = 10$) | **5876.86** | **13624.52** | **327.29** | **164.64** |
| KL-DASE | 4786.68 | 11036.75 | 236.26 | 119.42 |
| ES-DASE | 817.75 | 8328.72 | -7.57 | 65.63 |

*Table 5.* Ablation results when the replay size is 1,000,000. Bold values represent the maximum for each environment. Scores represent the average return over all agents of DASE.

| METHOD | ANT | HALFCHEETAH | LUNARLANDERCONTINUOUS | SWIMMER |
|---|---|---|---|---|
| DASE ($K = 2$) | 5434.32 | 12086.72 | 277.20 | 93.81 |
| DASE ($K = 5$) | 5921.87 | 12826.30 | 297.88 | 100.43 |
| DASE ($K = 10$) | **6217.24** | **13674.15** | **318.34** | **107.22** |
| KL-DASE | 5094.35 | 11075.49 | 227.78 | 77.25 |
| ES-DASE | 865.57 | 8354.15 | 0.34 | 64.17 |

We perform ablation studies to analyze the effects of the components: the usage of JS-divergence, number of agents, and off-policy correction (DPS). For this, we compare ablation over DASE under $K = \{2, 5, 10\}$, DASE with KL-divergence (KL-DASE), DASE without off-policy correction (no DPS is applied), and only with experience sharing (ES-DASE). As DASE is orthogonal to any off-policy deterministic policy gradient algorithm, ablation studies are performed under the TD3 algorithm (Fujimoto et al., 2018). Ablation results are given in Table 4 and 5, where average return over the last 10 evaluations over 10 trials of 1 million time steps is reported. Learning curves for the ablation studies can be found in our repository[1].

The complete algorithm outperforms every combination except when the number of agents increases. As the off-policy samples by other agents are safely corrected, the increasing number of agents yields more diverse exploration and thus slightly higher returns and faster convergence. However, training more agents linearly increases the training duration. We then replace the JS-divergence with KL-divergence (KL-DASE) in DPS. We obtain higher returns with JSD due to the symmetric measurement of the policies. Directed similarity measurement slightly degrades the algorithm's performance as two policies in the same environment cannot be completely distinct. Finally, we remove the off-policy correction in learning from other agents' experiences. The algorithm cannot converge and exhibits randomness in the action selection. Our theoretical approach is reflected empirically, that is, extrapolation error (Fujimoto et al., 2019) prevents agents from converging high evaluation returns due to the mismatch between the distributions under the agent's policy and samples collected by other agents.