Neighborhood Mixup Experience Replay: Local Convex Interpolation for Improved Sample Efficiency in Continuous Control Tasks

Ryan Sander[1] RMSANDER@MIT.EDU
Wilko Schwarting[1] WILKOS@MIT.EDU
Tim Seyde[1] TSEYDE@MIT.EDU
Igor Gilitschenski[2,3] GILITSCHENSKI@CS.TORONTO.EDU
Sertac Karaman[4] SERTAC@MIT.EDU
Daniela Rus[1] RUS@MIT.EDU
[1]*CSAIL - MIT,* [2]*University of Toronto,* [3]*Toyota Research Institute,* [4]*LIDS - MIT*

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

Abstract

Experience replay plays a crucial role in improving the sample efficiency of deep reinforcement learning agents. Recent advances in experience replay propose using Mixup (Zhang et al., 2018) to further improve sample efficiency via synthetic sample generation. We build upon this technique with Neighborhood Mixup Experience Replay (NMER), a geometrically-grounded replay buffer that interpolates transitions with their closest neighbors in state-action space. NMER preserves a locally linear approximation of the transition manifold by only applying Mixup between transitions with vicinal state-action features. Under NMER, a given transition's set of state-action neighbors is dynamic and episode agnostic, in turn encouraging greater policy generalizability via inter-episode interpolation. We combine our approach with recent off-policy deep reinforcement learning algorithms and evaluate on continuous control environments. We observe that NMER improves sample efficiency by an average 94% (TD3) and 29% (SAC) over baseline replay buffers, enabling agents to effectively recombine previous experiences and learn from limited data.

1. Introduction

Learning robust and effective behavior from a limited set of examples is a hallmark of human cognition (Fong et al., 2018). The sample efficiency of our neuronal circuits allows us to quickly learn new skills even with limited experience. In many problem domains, human sample efficiency far outperforms that of deep reinforcement learning algorithms (Lee et al., 2019). Narrowing this gap is a critical milestone toward replicating intelligence in reinforcement learning agents.

Model-free (MF), off-policy deep reinforcement learning (DRL) agents provide significant sample efficiency gains relative to their on-policy counterparts (Gu et al., 2017; Arulkumaran et al., 2017). Improved sample efficiency is due largely to experience replay techniques (Mnih et al., 2013), which enable agents to learn from past experience. The trial-and-error nature of reinforcement learning nonetheless necessitates collecting large volumes of training data (Yang et al., 2020; Yu, 2018; Mankowitz et al., 2019). While lower sample efficiency may be acceptable for learning in simulation, it may significantly hinder an agent's progress in many real-world applications where samples are expensive to generate (Yang et al., 2020).

Model-based (MB) DRL agents achieve improved sample efficiency by learning a model of the environment (Weyand et al., 2016; Ha and Schmidhuber, 2018; Hafner et al., 2020) that can be used for offline planning and policy refinement (Browne et al., 2012; Schwarting et al., 2021; Seyde et al., 2020, 2022). For reinforcement learning tasks with noisy and high-dimensional state and action spaces, however, an agent's learned environment models may suffer from estimation bias when little data is available. Combined with model capacity limitations, this can result in model-based agents converging to suboptimal policies (Mankowitz et al., 2019; Renaudo et al., 2015).

We aim to combine the benefits of learning on true environment interactions in MF-DRL with the sample efficiency benefits of MB-DRL. To this end, we propose Neighborhood Mixup Experience Replay (NMER), a modular replay buffer that improves the sample efficiency of off-policy, MF-DRL agents by training on experiences sampled from convex, linear combinations of vicinal transitions from the replay buffer. NMER interpolates neighboring pairs of transitions in the geometric transition space of the replay buffer using Mixup (Zhang et al., 2018), a convex and stochastic linear interpolation technique. Despite the computational and analytical simplicity of Mixup, its use in experience replay can improve generalization and policy convergence through implicit regularization and expansion of the training support of the agent's neural network function approximators. We empirically observe these benefits of Mixup in our continuous control experiments.

NMER can be applied to any continuous control reinforcement learning agent leveraging experience replay. As a motivating example, consider a robotic humanoid learning to walk using off-policy, MF-DRL agents, given a limited set of experiences consisting of odometry and actuator sensor measurements. With standard experience replay (Mnih et al., 2013) approaches, the finite size of the agent's experiences can hinder the agent from learning robust policies, perhaps due to the agent not experiencing a crucial subset of the transition space. With NMER, however, interpolated experiences can provide this DRL agent with crucial training samples in regions of the transition space they previously did not experience, thus improving the robustness and performance of the policies the agent learns. Our contributions[1] are thus summarized as follows:

1. **Neighborhood Mixup Experience Replay (NMER)**: A geometrically-grounded replay buffer that improves the sample efficiency of off-policy, MF-DRL agents by training these agents on linear combinations of vicinal transitions.

2. **Local Mixup**: A generalization of NMER, this algorithm considers the application of Mixup between vicinal points in any feature space, with proximity defined by a distance metric.

3. **Improved sample efficiency in continuous control**: Our evaluation study demonstrates that NMER substantially improves sample efficiency of off-policy, MF-DRL algorithms across several continuous control environments.

2. Related work

Experience replay, data augmentation, and interpolation approaches have been applied to RL and other machine learning domains. NMER builds off of these techniques to improve sample efficiency.

Experience replay *Prioritized Experience Replay* (PER) (Schaul et al., 2016) samples an agent's experiences from a replay buffer according to the "learnability" or "surprise" that each sample induces in the agent in its current parameterization. PER uses absolute TD-error of a sample (Schaul

1. Code for NMER can be found at this GitHub repository.

et al., 2016) as a heuristic measure of "surprise". In the stochastic prioritization variant of PER, transitions are sampled proportionally to their learnability. While this technique improves the sample efficiency by selecting highly-relevant samples, it does not improve the overall "learnability" of the samples themselves and restricts training to previously observed experience. *Experience Replay Optimization* (ERO) (Zha et al., 2019) parameterizes the replay buffer directly as a learned priority score function. Rather than using heuristics such as TD-error to determine a prioritization of samples, as is performed in PER (Schaul et al., 2016), in ERO this prioritization is learned directly via a policy gradient approach in which return is measured by the agent's policy improvement (Zha et al., 2019). A REINFORCE-based (Williams, 1992) estimate of the policy gradient updates the learned replay buffer in an alternating fashion with the policy being trained. Similarly to PER, while ERO improves sample efficiency by selecting samples with high "learnability", it does not improve the overall "learnability" of the samples themselves, and also restricts training to the agent's observed experiences. *Interpolated Rewards Replay* (von Pilchau et al., 2020) performs linear interpolation of experienced rewards. In contrast, NMER interpolates entire transitions using stochastic convex linear interpolation, resulting in a more expressive interpolation of an agent's experiences.

Data augmentation Data augmentation techniques are also used to improve the performance of DRL agents. *Reinforcement learning with Augmented Data (RAD)* is a module designed for improving agent performance in visual and propioceptive DRL tasks (Laskin et al., 2020). For continuous control environments, RAD leverages techniques such as random amplitude scaling (RAS). While RAS does allow for learning beyond an agent's observed set of experiences, it does not consider meaningful combinations of these experiences. In *Data-regularized Q (DrQ)* learning, geometric-invariant data augmentation mechanisms are applied to off-policy DRL algorithms to improve sample efficiency in visual control tasks, providing off-policy agents with sample efficiency comparable to state-of-the-art MB-DRL algorithms (Kostrikov et al., 2020). Similar to DrQ, NMER improves sample efficiency via regularization through training agents on augmented samples.

Mixup sampling Mixup was originally applied to supervised machine learning domains, and empirically improves the generalizability and out-of-sample predictive performance of learners (Zhang et al., 2018). Several reinforcement learning methodologies make use of Mixup-interpolated experiences for training reinforcement learning agents. In *Continuous Transition* (Lin et al., 2020), temporally-adjacent transitions are interpolated with Mixup, generating synthetic transitions between pairs of consecutive transitions. In *MixReg* (Wang et al., 2020), generated transitions are formed using Mixup on combinations of input and output signals. In *S4RL* (Sinha et al., 2021), generated transitions are produced by interpolating current (s_t) and next (s_{t+1}) states within an observed transition. While these approaches increase the training domain via interpolation, they do not strictly enforce geometric transition proximity of the resulting samples. Proximity between the points used for sampling is encoded temporally, as in (Lin et al., 2020; Sinha et al., 2021), but not in the geometric transition space of the agent's experience. NMER employs a nearest neighbor heuristic to encourage transition pairs for Mixup to be located approximately within the same dynamics regimes in the transition manifold. Compared to Continuous Transition (Lin et al., 2020) and S4RL (Sinha et al., 2021), samples interpolated with NMER may better preserve the local dynamics of the environment and enable further agent regularization through inter-episode interpolation between transitions and their dynamic sets of nearest neighbors.

3. Preliminaries

Neighborhood Mixup Experience Replay (NMER) builds on experience replay for off-policy DRL, Mixup, and nearest neighbor heuristics to encourage approximately on-manifold interpolation.

Off-policy DRL for continuous control tasks Off-policy DRL has successfully been applied to continuous control tasks through the use of actor-critic methods such as Soft Actor-Critic (SAC), Deep Deterministic Policy Gradients (DDPG), and Twin Delayed DDPG (TD3) (Haarnoja et al., 2018; Lillicrap et al., 2016; Fujimoto et al., 2018). In this off-policy, MF-DRL setting, agents are trained using transitions composed of states, actions, rewards, and next states. The state ($\mathcal{S}$), action ($\mathcal{A}$), and reward ($\mathcal{R}$) spaces that define these transitions are continuous.

Experience replay. Experience replay (Mnih et al., 2013) enables an agent to train on past observations. It can be largely decoupled from the agent's training algorithm - while the agent seeks to learn optimal policies and value functions given observed training samples, regardless of the samples provided to it, the experience replay buffer is tasked with providing the agent samples that offer the greatest "learnability" for improving these policies and value functions. Current experience replay approaches are discussed in Section 2.

Mixup. Mixup (Zhang et al., 2018) is a novel stochastic data augmentation technique that improves the generalizability of supervised learners by training them on convex linear combinations of existing samples. This linear interpolation mechanism invokes a prior on the learner that linear combinations of features result in the same linear combinations of targets (Zhang et al., 2018), leading to generalizability in accordance with Occam's Razor (Rasmussen and Ghahramani, 2001). Through another lens, Mixup increases the generalizability of a learner by increasing its training support with interpolated transitions. These transitions are sampled from the convex unit line connecting two transitions (Zhang et al., 2018) according to a symmetric beta distribution parameterized by α, which controls the spread of the distribution along this relative unit line. To interpolate a new sample $\mathbf{x}_{\text{interpolated}}$ using two existing samples $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$, Mixup interpolates according to:

$$\mathbf{x}_{\text{interpolated}} = \lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2, \ \lambda \sim \beta(\alpha, \alpha), \ \alpha > 0 \tag{1}$$

On-manifold interpolation. To measure the accuracy of interpolation in interpolated experience replay approaches, we consider how "on-manifold" the interpolated transition is with respect to the transition manifold mapping states and actions to rewards and next states. We consider that the observed transitions of a replay buffer lie on a transition manifold, which we denote by the space $\mathcal{T}$. The space $\mathcal{T}$ is given by the Cartesian product of the state ($\mathcal{S}$), action ($\mathcal{A}$), reward ($\mathcal{R}$), and next state ($\mathcal{S}$) spaces of the agent as: $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S}$. With NMER, the use of Mixup between only proximal/neighboring transitions acts as a heuristic to encourage interpolated transitions to remain near the underlying transition manifold, as depicted by Figure 1. See our technical report for details on on-manifold assessment.

4. Neighborhood Mixup Experience Replay (NMER)

NMER trains off-policy MF-DRL agents using convex linear combinations of an agent's existing, proximal experiences, effectively creating locally linear models centered around each transition of the replay buffer. By only interpolating proximal transitions with one another, where proximity is measured by the standardized Euclidean distance in the state-action space of the replay buffer,

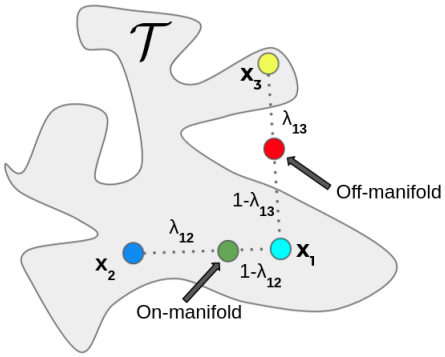

Figure 1: Examples of on and off-transition manifold interpolation using Mixup. On-manifold or approximately on-manifold interpolation is crucial for successfully training DRL agents in continuous control tasks.

NMER interpolates transitions that have similar state and action inputs, but potentially different reward and next state outputs. In considering these nearest neighbors, NMER regularizes the off-policy MF-DRL agents it trains by allowing inter-episode interpolation between proximal transitions. Furthermore, in the presence of stochasticity in the transition manifold, NMER can prevent these agents from overfitting to a particular (reward, next state) outcome by interpolating different (reward, next state) outcomes for near-identical (state, action) inputs. NMER consists of two steps:

1. **Update step**: When a new environment interaction is added to the replay buffer, re-standardize the states and actions of the stored transitions in the replay buffer, and update the nearest neighbor data structures using Euclidean distances over the Z-score standardized, concatenated state-action features of the replay buffer. Similarity search is thus measured over the input state and action spaces; however, we emphasize that NMER can admit other distance functions and representations of similarity as well. See our technical report for further details.

2. **Sampling Step**: First, we sample a batch of "sample transitions" uniformly from the replay buffer. Next, we query the nearest neighbors of each transition in this sampled batch. Following this, for each set of neighbors in the training batch, we sample a neighbor transition uniformly from this set of neighbors, and apply Mixup to linearly interpolate each pair of selected samples and neighbors ($\mathbf{x}_{\text{sample},i}$ and $\mathbf{x}_{\text{neighbor},i}$, respectively):

$$\mathbf{x}_{\text{interpolated},i} = \lambda \mathbf{x}_{\text{sample},i} + (1-\lambda)\mathbf{x}_{\text{neighbor, i}}, \;\; \lambda \sim \beta(\alpha, \alpha), \alpha > 0 \tag{2}$$

These steps are given in Algorithm 1, and depicted in Figure 2. NMER introduces minimal computational overhead compared to standard experience replay, requiring only vectorized standardization, nearest neighbor querying, and local Mixup operations. This positions NMER as a viable experience replay buffer for high-dimensional continuous control tasks.

Agent regularization via linear interpolation. Through the lens of Occam's Razor (Rasmussen and Ghahramani, 2001), NMER improves the generalizability of the policy and value function approximators of off-policy, MF-DRL agents by invoking the prior that linear combinations of state-action pairs result in the same linear combinations of corresponding reward-next state pairs. This prior improves generalizability in tasks where this linearity assumption approximately holds. Since the spaces of an agent in continuous control tasks are continuous, interpolating continuous, linear

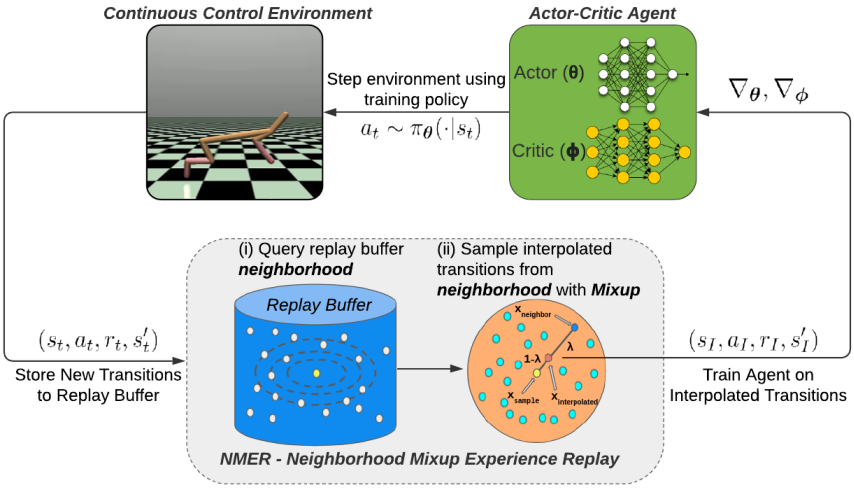

Figure 2: With NMER, convex interpolation is performed between a sampled transition and its neighboring transitions, improving the generalizability and robustness of off-policy, MF-DRL agents applied to continuous control tasks.

Algorithm 1 Neighborhood Mixup Experience Replay (NMER)

Input: Replay buffer $\mathcal{B}$, Mixup hyperparameter $\alpha > 0$, Batch Size T
Output: Interpolated training batch $\mathbf{B}_{\text{train}}$

$\mathbf{B} = \{(s_t, a_t, r_t, s'_t)\}_{t=1}^{\text{T}} \overset{\text{iid}}{\sim} \mathcal{U}(\mathcal{B})$ // Sample Batch uniformly from Replay Buffer
$\mathbf{B}_{\text{train}} \leftarrow \text{Array}[\cdots]$
for t in T **do**
  $(s_s, a_s, r_s, s'_s) \leftarrow \mathbf{B}[t]$ // Sampled Transition for NMER
  $[\tilde{s}_s, \tilde{a}_s]^T \leftarrow \textbf{ZScore}([s_s, a_s]^T)$ // Standardize States and Actions of Sampled Transition
  $K_s \leftarrow \textbf{NN}\left([\tilde{s}_s, \tilde{a}_s]^T, \mathcal{B}\right)$ // Standardized Local Neighborhood of Sampled Transition
  $(s_n, a_n, r_n, s'_n) \sim \mathcal{U}(K_s)$ // Sample Neighboring Transition from Local Neighborhood
  $\lambda \sim \beta(\alpha, \alpha)$ // Sample Mixup Coefficient
  $\mathbf{x}_s \leftarrow [s_s, a_s, r_s, s'_s]^T$ // Sampled Transition Features
  $\mathbf{x}_n \leftarrow [s_n, a_n, r_n, s'_n]^T$ // Neighboring Transition Features
  $\mathbf{x}_i = \lambda \mathbf{x}_s + (1 - \lambda)\mathbf{x}_n$ // Interpolate Sampled and Neighboring Transitions using Mixup
  $\mathbf{B}_{\text{train}}[t] \leftarrow \mathbf{x}_i$ // Add Interpolated Sample to Training Batch
end for
return $\mathbf{B}_{\text{train}}$

combinations of transitions can still yield interpolated samples that lie proximate to the underlying transition manifold $\mathcal{T}$.

Furthermore, if the transition manifold $\mathcal{T}$ is convex, NMER guarantees on-manifold interpolation, since this technique generates strictly convex combinations of transitions. In this regime, synthetically-generated on-manifold transitions are indistinguishable from transitions generated at the same point using the underlying environment dynamics. However, for many applications, par-

ticularly high-dimensional, real-world continuous control tasks, the underlying transition manifold will generally be non-convex.

Neighborhood Mixup as a heuristic to encourage on-manifold interpolation. Non-convexity and nonlinearity in continuous control environments provide motivation for our neighborhood-based interpolation mechanism, which addresses issues with non-convexity of the transition manifold by only considering interpolation between transitions in the same "neighborhood", i.e. transitions with similar state-action pairs. If the transition manifold is locally Euclidean, linearly interpolating two transitions is a suitable, approximately on-manifold mechanism for interpolating between spatially proximal transitions.

5. Continuous control evaluation

To rigorously evaluate and quantify the improvement in sample efficiency with NMER, we compare NMER to other state-of-the-art replay buffers by applying these replay buffers to continuous control tasks and off-policy, MF-DRL algorithms.

Testing environment and configuration. We consider continuous control environments from the OpenAI Gym MuJoCo (Brockman et al., 2016; Todorov et al., 2012) suite. Since we are principally interested in evaluating the sample efficiency of replay buffers, we treat the off-policy, MF-DRL algorithms used for these evaluations (SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018)) as part of the experimental configuration. For each replay buffer variant, including NMER, we train agents using replay or update-to-data ratios (ratio of gradient steps to environment interactions) of 1, 5, and 20, and report the best results for each replay buffer variant. Additionally, for SAC (Haarnoja et al., 2018), to stabilize the policy, we add a small L2 regularizer to the actor network for all SAC evaluations. Implementation details and ablation studies for each replay buffer variant are provided in the technical report. We measure replay buffer sample efficiency using the evaluation reward of the reinforcement learning agent after 200K environment interactions have been sampled, as in (Lin et al., 2020; Lee et al., 2020). Rewards are smoothed using an averaging window of 11, as in (Achiam, 2018).

Baselines. We compare NMER to the following baselines: (i) *Uniform, Vanilla Replay* (`U`) (Mnih et al., 2013; Engel et al., 2005), where transitions are sampled i.i.d. uniformly from the replay buffer, (ii) *Prioritized Experience Replay* (`PER`) (Schaul et al., 2016) with stochastic prioritization, (iii) *Continuous Transition* (`CT`) (Lin et al., 2020). Since the main comparison between NMER and CT is how samples are selected for interpolation, we make two modifications to the original CT baseline: (a) We remove the automatic Mixup α hyperparameter tuning mechanism, and (b) If a terminal state is encountered in either the sample or neighbor transition, no interpolation occurs, and the sampled transition is simply used for training the agent. In Continuous Transition, terminal transitions in an episode can be interpolated with their previous, non-terminal transitions, resulting in non-binary termination signals. In order to make the fairest comparison possible between NMER and CT, we adopt the same interpolation rules we use for NMER in CT. (iv) *SUNRISE Baselines*. Additionally, we compare the performance of NMER to SUNRISE baselines, tested using the same continuous control environments and 200K environment interaction evaluation (Lee et al., 2020). (v) *Neighborhood Size Limits of NMER* (`1NN-Mixup` and `Mixup`, respectively), which represent the limits of NMER with one neighbor ($k = 1$) and all neighbors ($k = |\mathcal{B}|$), respectively. (vi) *S4RL* (`S4RL`), which implements the replay buffer interpolation technique from (Sinha et al., 2021).

Lastly, (vii) *Noisy Replay* ($\mathcal{N}(0, \sigma^2)$), which trains DRL agents on transitions with i.i.d. Gaussian noise added to each component.

Results for these evaluations are provided in Tables 1, 2, and 3. Note the following abbreviations in Tables 1 and 3: (i) `U` = Uniform Replay, (ii) `PER` = Prioritized Experience Replay, (iii) `CT` = Continuous Transition, (iv) `NMER` = Neighborhood Mixup Experience Replay, (v) `1-NN Mixup` = NMER with one neighbor, (vi) `Mixup` = NMER with all neighbors (Naive Mixup), (vii) `S4RL` = S4RL, and (viii) $\mathcal{N}(0, \sigma^2)$ = Noisy Replay. Best results for each off-policy algorithm (`TD3`, `SAC`) are bolded. Detailed and additional results can be found in our technical report. Learning curves for `TD3` and `SAC` agents across all evaluated replay buffers are depicted in Figures 3 and 4 (`SUNRISE` and other baselines evaluated in (Lee et al., 2020) are depicted with dashed horizontal lines indicating mean evaluation reward at 200K environment interactions). Each result is averaged over four runs.

The results of this continuous control evaluation study indicate that NMER frequently achieves comparatively better sample efficiency than the baseline replay buffers used in this study across SAC and TD3, as well as other baseline DRL algorithms evaluated in (Lee et al., 2020).

Table 1: Continuous control results from OpenAI Gym MuJoCo, 200K env. interactions.

RL Agent	**Environment**	U	PER	CT	NMER
TD3	Ant	2005 ± 399	2317 ± 756	2834 ± 875	**4347 ± 908**
	HalfCheetah	6467 ± 658	6447 ± 693	8097 ± 358	**9340 ± 1678**
	Hopper	3252 ± 157	3213 ± 511	3156 ± 351	**3393 ± 220**
	Swimmer	131 ± 20	**138 ± 8**	134 ± 10	122 ± 10
	Walker2d	2236 ± 686	1452 ± 1057	3087 ± 1058	**4611 ± 441**
	Humanoid	388 ± 66	860 ± 385	2242 ± 2027	**4930 ± 190**
	Δ NMER (%)	-37.5%	-36.8%	-22.1%	0%
SAC	Ant	1188 ± 692	800 ± 160	1594 ± 717	**2721 ± 1685**
	HalfCheetah	4918 ± 1928	6880 ± 886	6120 ± 525	**8168 ± 1585**
	Hopper	1692 ± 1160	**2801 ± 827**	1115 ± 897	1875 ± 900
	Swimmer	110 ± 29	121 ± 42	106 ± 46	**140 ± 10**
	Walker2d	4303 ± 636	3466 ± 784	**4696 ± 1194**	4429 ± 819
	Δ NMER (%)	-26.0%	-14.4%	-25.1%	0%

Table 2: Baselines from SUNRISE (Lee et al., 2020) vs TD3 + NMER, 200K env. interactions.

Environment	METRPO	PETS	POPLIN-A	POPLIN-P	SUNRISE	NMER+TD3
Ant	282±18	1166±227	1148±438	2330±321	1627±293	**4347±908**
HalfCheetah	2284±900	2288±1019	1563±1137	4235±1133	5371±483	**9340±1678**
Hopper	1273±501	115±621	203±963	2055±614	2602±307	**3393±220**
Walker2d	-1609±658	283±502	-105±250	597±479	1926±695	**4611±441**
Δ NMER (%)	-82.8%	-84.8%	-87.7%	-56.9%	-46.7%	0%

Table 3: Additional baseline comparison study with TD3, 200K env. interactions.

Environment	1NN-Mixup	Mixup	S4RL	$\mathcal{N}(0, \sigma^2)$	NMER
Ant	2651 ± 828	2361 ± 616	769 ± 270	1709 ± 350	$\mathbf{4347 \pm 908}$
HalfCheetah	7255 ± 1014	6743 ± 657	6032 ± 187	3733 ± 540	$\mathbf{9340 \pm 1678}$
Hopper	3360 ± 217	2917 ± 523	960 ± 151	1287 ± 593	$\mathbf{3393 \pm 220}$
Swimmer	111 ± 22	43 ± 5	44 ± 7	39 ± 1	$\mathbf{122 \pm 10}$
Walker2d	3372 ± 833	3340 ± 293	514 ± 169	516 ± 157	$\mathbf{4611 \pm 441}$
Δ NMER (%)	-22.9%	-36.0%	-68.4%	-67.9%	0%

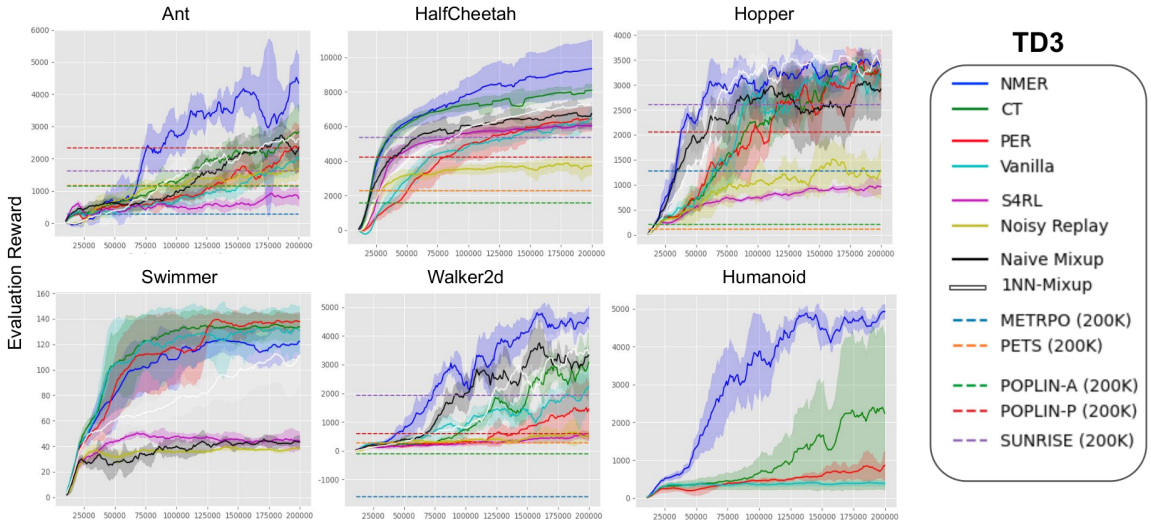

Figure 3: Learning curves for TD3 agents trained on NMER and baselines. Each replay buffer is run with four random seeds, and we plot mean performance with $\pm 1\sigma$ intervals.

6. Discussion

These evaluation studies demonstrate that agents trained using NMER can learn robust policies using fewer environment interactions compared to agents trained using state-of-the-art replay buffers for continuous control. We consider the implications of NMER and its extensions to DRL.

Limitations. Despite the observed empirical success, NMER exhibits several limitations. NMER assumes the underlying transition dynamics are locally linear, which may be a naive approximation for continuous control tasks with nonlinear underlying dynamics. Additionally, although NMER's use of nearest neighbors serves as a viable heuristic to steer the replay buffer toward on-manifold interpolation, on-manifold interpolation is not analytically guaranteed. The sections below aim to put these limitations into context by suggesting viable generalizations and considerations for future work. Third, our experiments illustrate the limits of high replay (update-to-data) ratios, which could be explained by overestimation bias of the value function(s) (Chen et al., 2020; Hiraoka et al., 2021).

Generalizing Synthetic Training. As NMER uses convex interpolation to generate transitions for training, as more samples are added to the replay buffer, the interpolated transitions will become more accurate. Furthermore, NMER can be extended to modulate the ratio of real to synthetic

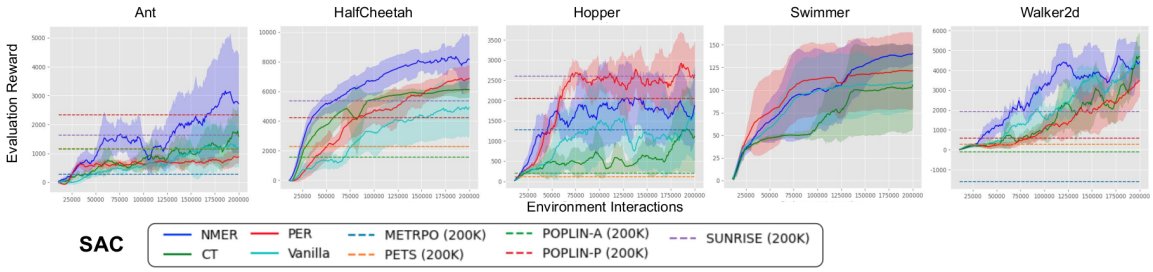

Figure 4: Learning curves for SAC agents trained on NMER and baselines. Each replay buffer is run with four random seeds, and we plot mean performance with $\pm 1\sigma$ intervals.

training samples over time, enabling a variety of flexible training schemes for different replay buffer densities.

Generalizing neighborhoods. NMER computes nearest neighbors using standardized Euclidean norms over concatenated state-action space, which allows for Mixup-based interpolation of transitions with proximal state-action vectors, regardless of the rewards and next states in these transitions. However, this notion of proximity between transitions can be generalized to any measure of distance in the transition space of a replay buffer. Generalizing this notion of proximity between stored environment interactions can be invoked via different distance metrics, e.g. Mahalanobis distance, as well as the use of composite product norms over different features in the transition space. For instance, the use of a composite product norm over states $\times$ actions results in nearest neighbors having similar (state, action) pairs. The efficacy of different proximity representations for neighborhood-based interpolated experience replay remains an open research question.

Generalizing interpolation. NMER also invokes the implicit prior that linear combinations of states and actions result in the same linear combination of rewards and next states through the use of Mixup. However, under some dynamics regimes and continuous control environments, local linear interpolation via local Mixup may result in interpolated samples far from the underlying transition manifold. For interpolating on-manifold samples in locally nonlinear neighborhoods, off-policy DRL agents may benefit from the use of more sophisticated neighborhood-based interpolation mechanisms, such as Gaussian Process Regression (Rasmussen, 2003; Rasmussen and Kuss, 2004) or Graph Neural Networks (Zhou et al., 2020).

7. Conclusion

We present Neighborhood Mixup Experience Replay (NMER), an experience replay buffer that improves the sample efficiency of off-policy DRL agents through synthetic sample generation. We empirically demonstrate that training agents on experiences generated via local Mixup in the transition space of a replay buffer facilitates learning robust policies using fewer environment interactions. NMER combines the benefits of learning from locally linear approximations of the underlying environment model with the sample efficiency benefits of learning from synthetic samples, thus expanding the possibilities for tractable DRL in real-world continuous control settings.

Acknowledgments

This research was supported by the Toyota Research Institute (TRI). This article solely reflects the opinions and conclusions of its authors and not TRI, Toyota, or any other entity. We thank TRI for their support. The authors thank the MIT SuperCloud and Lincoln Laboratory Supercomputing Center (Reuther et al., 2018) for providing HPC and consultation resources that have contributed to the research results reported within this publication.

References

Joshua Achiam. Spinning up in deep reinforcement learning. 2018.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2020.

Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 201–208, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102377.

Ruth C Fong, Walter J Scheirer, and David D Cox. Using human brain activity to guide machine learning. *Scientific reports*, 8(1):1–10, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2455–2467, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 7 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. *arXiv preprint arXiv:2110.02034*, 2021.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19884–19895. Curran Associates, Inc., 2020.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. *arXiv preprint arXiv:2007.04938*, 2020.

Su Young Lee, Choi Sungik, and Sae-Young Chung. Sample-efficient deep reinforcement learning via episodic backward update. In *Advances in Neural Information Processing Systems*, pages 2112–2121, 2019.

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. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

Junfan Lin, Zhongzhan Huang, Keze Wang, Xiaodan Liang, Weiwei Chen, and Liang Lin. Continuous transition: Improving sample efficiency for continuous control problems via mixup. *arXiv preprint arXiv:2011.14487*, 2020.

Daniel J. Mankowitz, Gabriel Dulac-Arnold, and Todd Hester. Challenges of real-world reinforcement learning. 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems*, pages 751–759, 2004.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. 2001.

Erwan Renaudo, Benoît Girard, Raja Chatila, and Mehdi Khamassi. Respective advantages and disadvantages of model-based and model-free reinforcement learning in a robotics neuro-inspired cognitive architecture. *Procedia Computer Science*, 71:178–184, 2015. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2015.12.194. 6th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2015, 6-8 November Lyon, France.

Albert Reuther, Jeremy Kepner, Chansup Byun, Siddharth Samsi, William Arcand, David Bestor, Bill Bergeron, Vijay Gadepally, Michael Houle, Matthew Hubbell, et al. Interactive supercomputing on 40,000 cores for machine learning and data analysis. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2018.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR (Poster)*, 2016.

Wilko Schwarting, Tim Seyde, Igor Gilitschenski, Lucas Liebenwein, Ryan Sander, Sertac Karaman, and Daniela Rus. Deep latent competition: Learning to race using visual control policies in latent space. *Proceedings of the 2020 Conference on Robot Learning, PMLR 155:1855-1870*, 2021.

Tim Seyde, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Learning to plan via deep optimistic value exploration. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger, editors, *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pages 815–825, The Cloud, 6 2020. PMLR.

Tim Seyde, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Learning to plan optimistically: Uncertainty-guided deep exploration via latent model ensembles. In *Conference on Robot Learning*, pages 1156–1167. PMLR, 2022.

Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4RL: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *5th Annual Conference on Robot Learning*, 2021.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Wenzel Baron Pilar von Pilchau, Anthony Stein, and Jörg Hähner. Bootstrapping a dqn replay memory with synthetic experiences. *arXiv preprint arXiv:2002.01370*, 2020.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7968–7978. Curran Associates, Inc., 2020.

Tobias Weyand, Ilya Kostrikov, and James Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1–10. PMLR, 11 2020.

Yang Yu. Towards sample efficient reinforcement learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5739–5743. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/820.

Daochen Zha, Kwei-Herng Lai, Kaixiong Zhou, and Xia Hu. Experience replay optimization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4243–4249. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/589.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.