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

Ryan Sander*
MIT CSAIL
rmsander@mit.edu

Wilko Schwarting
Isee, MIT CSAIL
wilkos@mit.edu

Tim Seyde
MIT CSAIL
tseyde@mit.edu

Igor Gilitschenski
University of Toronto, MIT CSAIL
gigor@mit.edu

Sertac Karaman
MIT LIDS
karaman@mit.edu

Daniela Rus
MIT CSAIL
rus@mit.edu

Abstract

The human brain is remarkably sample efficient, capable of learning complex behaviors by meaningfully combining previous experiences to simulate novel ones, even when few experiences are available. To improve sample efficiency in continuous control tasks, we take inspiration from this learning phenomenon. We propose Neighborhood Mixup Experience Replay (NMER), a modular replay buffer that interpolates transitions with their closest neighbors in normalized state-action space. NMER preserves a locally linear approximation of the transition manifold by only interpolating transitions with similar 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 cross-episode interpolation. We combine our approach with recent off-policy reinforcement learning algorithms and evaluate on several continuous control environments. We observe that NMER improves sample efficiency by 40% over our baseline replay buffers, enabling agents to effectively recombine previous experience and learn from limited data.

1 Introduction

Learning robust and effective behavior from a limited set of examples is a hallmark of human cognition [6]. The sample efficiency of our brain and neuronal circuits allows us to quickly learn new skills, perform new tasks, and generalize the experience we observe to a variety of different environments and settings. In many problem domains, human sample efficiency far outperforms the sample efficiency of deep reinforcement learning algorithms [14]. Narrowing this gap is a critical milestone towards effectively replicating intelligence in reinforcement learning settings.

Model-free (MF), off-policy deep reinforcement learning (DRL) algorithms have revolutionized agents' abilities to learn policies for real-world continuous control tasks using fewer environment interactions than their on-policy MF-DRL counterparts [8, 2]. Improved sample efficiency of these off-policy agents is due in part to the application of experience replay [18] techniques, which enable for agents to learn from their previous experiences by training on them (thereby "replaying" these experiences). The trial-and-error nature of reinforcement learning nonetheless still necessitates the need to collect large volumes of training data [32, 33, 17]. While lower sample efficiency is acceptable for learning in simulation, as samples can be generated from the applicable simulator, for

*Correspondence to rmsander@mit.edu.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

many real-world continuous control tasks, lower sample efficiency can significantly hinder an agent's ability to learn if there are a limited number of environment interactions available for training [32].

Model-based (MB) DRL agents achieve improved sample efficiency by learning a model of the environment [30, 9, 11, 24], allowing for the use of simulation and sampling methods such as Monte Carlo Tree Search (MCTS) [4], latent imagination [11, 24], or Deep Optimistic Value Exploration (DOVE) [25] using this learned environment model. For reinforcement learning tasks with noisy and high-dimensional state and action spaces, however, an agent's learned environment models may suffer from estimation error and bias where little data is available [17, 22]. The presence of this bias, as well as model capacity limitations, can result in model-based agents converging to sub-optimal policies for the given task(s).

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 MF, off-policy DRL agents by training on experiences sampled from convex, locally linear interpolated transitions of the replay buffer. At a high level, NMER interpolates proximal pairs of transitions in the transition space of the replay buffer using Mixup [35], a convex and stochastic linear interpolation technique. Despite the computational and analytical simplicity of Mixup, which enables for runtime-efficient experience replay, the use of Mixup in experience replay can enable for better generalization and policy convergence in MF, off-policy DRL agents through implicit regularization and increasing the training support of the agent's function approximators. We empirically observe these beneficial effects of Mixup in our continuous control experiments.

NMER can be applied to any continuous control task, i.e. any task with continuous state, action, and reward spaces, as well as any reinforcement learning agent that uses a replay buffer. As a motivating example, consider a robotic humanoid learning to walk using MF, off-policy DRL, given a limited set of experiences consisting of odometry and actuator sensor measurements. With standard experience replay [18] 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 are summarized as follows:

1. **Neighborhood Mixup Experience Replay (NMER)**: A replay buffer that improves sample efficiency of off-policy DRL agents by training these agents on locally linear interpolated combinations of neighboring transitions.
2. **Local Mixup**: A generalization of NMER, this algorithm considers the application of Mixup between local points in any feature or vector space by only applying Mixup to nearest neighbors measured using a given distance metric. In highly nonlinear features spaces, local Mixup enables for learner generalization and regularization without interpolating too far across highly different dynamic regions of the feature space.
3. **Improved sample efficiency in continuous control**: Our evaluation study demonstrates that NMER substantially improves sample efficiency of MF, off-policy DRL algorithms across several continuous control environments.

2 Related work

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

Experience replay. *Prioritized Experience Replay* (PER) [23] 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. To estimate "learnability", PER uses absolute TD-error of a sample [23] as a heuristic measure of this quantity. In the stochastic prioritization variant of PER, this algorithm samples transitions from the replay buffer proportionally to the learnability of the samples. While this replay augmentation algorithm improves the sample efficiency of an agent by selecting

highly-relevant samples, it does not improve the overall "learnability" of the samples themselves, and still restricts the training domain of the agent to its set of observed experiences. *Experience Replay Optimization* (ERO) [34] 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 in the replay buffer, as is performed in PER [23], in ERO this prioritization is learned directly via a policy gradient approach in which return is measured by the agent's policy improvement [34]. A REINFORCE-based [31] 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 the training domain of the agent to its set of observed experiences. *Interpolated Rewards Replay* [28] performs linear interpolation of experienced rewards. In contrast, NMER interpolates entire transitions using stochastic, locally linear, convex interpolation, thus allowing for richer interpolation of an agent's experiences.

Data augmentation for reinforcement learning. Training data augmentation has been successfully used to improve performance of DRL agents. *Reinforcement learning with Augmented Data (RAD)* is a module designed for improving agent performance in visual and propioceptive DRL tasks [12]. For continuous control environments, such as the OpenAI Gym MuJoCo [3, 27] tasks considered in this paper, RAD leverages data augmentation techniques such as random amplitude scaling. While this does allow for learning beyond an agent's observed set of experiences, augmentations such as randomized amplitude scaling make no use of the other transitions stored in the replay buffer to determine amplitude scaling factors, and do not combine the existing transitions that agents store. In *Data-Regularized Q (DrQ)* learning, generalizable and image-invariant data augmentation mechanisms are applied to off-policy DRL algorithms to improve sample efficiency in pixels-to-control tasks, providing off-policy agents with sample efficiency comparable to state-of-the-art model-based reinforcement learning algorithms. Though DrQ is applied to pixels-to-control tasks and NMER is applied to state-space control tasks, both of these methodologies achieve improved sample efficiency via regularization through training agents on augmented samples.

Mixup for sample-efficient reinforcement learning. Mixup was originally applied to supervised and unsupervised machine learning domains, and empirically improves the generalizability and out-of-sample predictive performance of learners [35]. Several reinforcement learning methodologies make use of Mixup-interpolated experiences for training reinforcement learning agents. In *Continuous Transition* [16], temporally-adjacent transitions are interpolated with one another using Mixup, in effect creating a continuum of synthetic transitions between any two observed transitions. In *MixReg* [29], interpolated transitions are formed using Mixup on combinations of input and output RL signals. In *S4RL* [26], interpolated transitions are produced by interpolating current and next states within an observed transition. While these methodologies include sophisticated mechanisms for interpolating samples for training in continuous state and action spaces, these approaches do not strictly enforce transition locality of the interpolated samples produced. Proximity between the points used for sampling is encoded temporally, as in [16, 26], but not in the transition space of the agent's experience. NMER uses a nearest neighbor heuristic to ensure that transitions used for Mixup-based interpolation are located approximately within the same dynamics regimes of the transition manifold. Compared to Continuous Transition [16] and S4RL [26], samples interpolated with NMER can better preserve the local dynamics of the environment and enable further agent regularization through cross-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 Deep Deterministic Policy Gradients (TD3) [10, 15, 7]. In this off-policy, MF-DRL setting, agents are trained using transitions composed of states, actions, rewards, and next states. The spaces in which these transitions lie are continuous, i.e. are all subsets of $\mathbb{R}^d$.

Experience replay. Experience replay [18] is a technique used to improve the sample efficiency of off-policy MF-DRL and MB-DRL agents by training an agent on their past, or "replayed" experiences to improve the agent's policies and value functions. Experience replay 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 most "learnability" for improving these policies and value functions. Current experience replay approaches are discussed in Section 2.

Mixup. Mixup [35] is a novel stochastic data augmentation technique used to improve the generalizability of deep neural networks by training on convex linear combinations of existing points. Intuitively, this linear interpolation mechanism invokes the Bayesian linear prior on the learner that linear combinations of features result in the same linear combinations of targets [35], leading to generalizable out-of-sample performance in accordance with Occam's Razor [20]. Through another lens, Mixup aims to increase the generalizability of the model it is applied to by increasing the support of the model through training on interpolated examples. Mixup performs interpolation by sampling from the convex unit line connecting two samples [35] according to a beta distribution parameterized by a hyperparameter 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:

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

Where $\alpha > 0$ is a hyperparameter controlling the spread of the beta distribution.

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.

More rigorously, 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:

$$\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S} \tag{2}$$

With NMER, the use of local Mixup (convex) interpolation between only nearest neighbors acts as a heuristic to encourage interpolated samples to stay near the transition manifold. Figure 1 depicts on and off-transition manifold interpolation when generating interpolated transitions using Mixup.

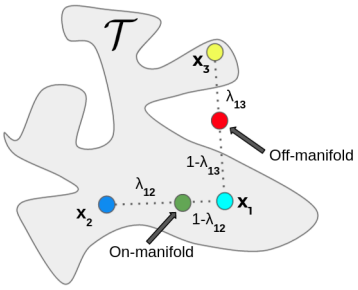

Figure 1: Examples of on and off-transition manifold interpolation using Mixup. On-manifold or approximately on-manifold interpolation is crucial for successfully training DRL agents in continuous control tasks.

4 Neighborhood Mixup Experience Replay (NMER)

NMER trains off-policy DRL agents using locally linear convex combinations of an agent's existing experiences, in effect 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, we consequently interpolate 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

reinforcement learning agents it trains by allowing for cross-episodic interpolation between proximal transitions. Furthermore, in the presence of stochasticity in the transition manifold, NMER can prevent RL agents from overfitting to a particular (reward, next state) outcome by interpolating different (reward, next state) outcomes for near-identical (state, action) inputs. With NMER, the agent learns to consider not only one combination of (reward, next state) pairs given an input (state, action) pair, but rather unbiased linear combinations of them, results in the agent learning the conditional expectation of rewards and next states given states and actions. For MF, off-policy DRL agents and continuous control tasks, the role of NMER can be visualized in Figure 2.

$$\mathbb{E}_{x\sim\mathcal{B}}\left[\mathbb{E}_{\lambda\sim\beta(\alpha,\alpha),x_{nn}\sim KNN(x)}\left[\lambda x+(1-\lambda)x_{nn}\right]\right]$$

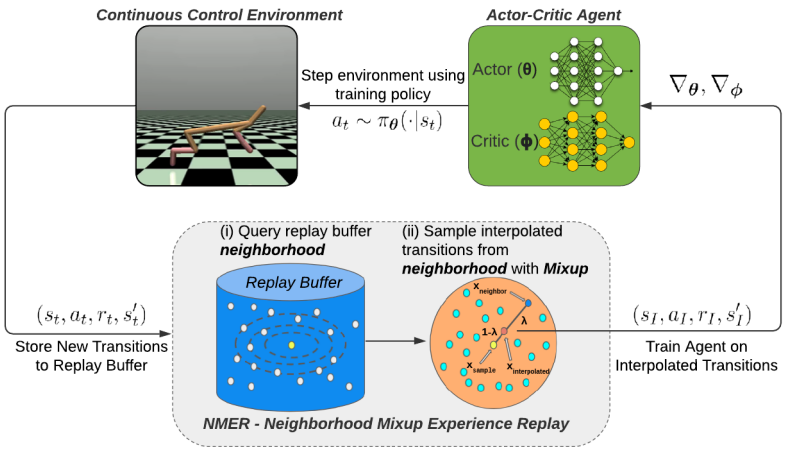

Figure 2: Neighborhood Mixup Experience Replay (NMER) applied in model-free, off-policy DRL. Effectively, NMER applies an additional step to the standard off-policy replay buffer. Local convex interpolation is achieved via the Mixup technique proposed in [35]. As NMER is conceptually similar to standard replay buffers, this allows for high modularity with NMER across model-free, off-policy reinforcement learning algorithms and continuous state and action space environments.

The steps of NMER are as follows:

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

$$\mathbf{x}_{\text{interpolated},i} = \lambda\mathbf{x}_{\text{sample},i} + (1-\lambda)\mathbf{x}_{\text{neighbor, i}},\ \lambda \sim \beta(\alpha,\alpha), \alpha > 0 \tag{3}$$

These steps are given in Algorithm 1. For run time efficiency, we implement this in a batched, vectorized fashion over the training batches. NMER introduces minimal computational overhead compared to vanilla experience replay - the only additional operations required are standardization, nearest neighbor querying, and local Mixup. This minimal additional overhead positions NMER as a viable experience replay buffer for continuous control environments of any complexity and dimensionality.

Agent regularization via linear interpolation. Through the lens of Occam's Razor [20], NMER improves the generalizability of the policy and value function approximators of the agent by invoking

Algorithm 1 Neighborhood Mixup Experience Replay (NMER)

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

$\mathbf{B} = \{(s_t, a_t, r_t, s'_t)\}_{t=1}^{T} \overset{\text{iid}}{\sim} \mathcal{U}(\mathcal{B})$ // Sample Batch Uniformly From Replay Buffer
$\mathbf{B}_{\text{train}} \leftarrow \text{Array}[\cdots]$
for t in T **do**
  $(s_s, a_s, r_s, s'_s) \leftarrow \mathbf{B}[t]$ // Sample Transition to Interpolate With
  $[\tilde{s}_s, \tilde{a}_s]^T \leftarrow \textbf{ZScore}([s_s, a_s]^T)$ // Standardize States and Actions
  $k_s \leftarrow \textbf{NN}\Big([\tilde{s}_s, \tilde{a}_s]^T, \mathcal{B}\Big)$ // Standardized Nearest Neighbors of Sample Point
  $(s_n, a_n, r_n, s'_n) \sim \mathcal{U}(k_s)$ // Sample Uniformly From Local Neighborhood
  $\mathbf{x}_s \leftarrow [s_s, a_s, r_s, s'_s]^T$ // Sample Transition Features
  $\mathbf{x}_n \leftarrow [s_n, a_n, r_n, s'_n]^T$ // Neighbor Transition Features
  $\lambda \sim \beta(\alpha, \alpha)$ // Sample Mixup Coefficient Using Hyperparameter α
  $\mathbf{x}_i = \lambda \mathbf{x}_s + (1 - \lambda)\mathbf{x}_n$ // Interpolate Sample and Neighbor Transitions Using Mixup
  $\mathbf{B}_{\text{train}}[t] \leftarrow \mathbf{x}_i$ // Add Interpolated Sample to Training Batch
end for
return $\mathbf{B}_{\text{train}}$

the prior that linear combinations of state-action pairs result in the same linear combinations of corresponding reward-next state pairs. This prior, although strong in some continuous control settings, can enable improved generalizability in tasks where this linearity assumption approximately holds. Since the spaces of an agent in continuous control tasks are continuous, interpolating continuous, linear combinations of transitions can still yield interpolated samples that lie on, or approximately on, the underlying transition manifold $\mathcal{T}$.

Furthermore, if the transition manifold $\mathcal{T}$ is convex, NMER allows for guaranteed on-manifold interpolation; since NMER only generates convex combinations of experiences, interpolated transitions will be guaranteed to lie on the transition manifold if it is convex. Crucially, synthetically-generated on-manifold transitions are indistinguishable from transitions generated at the same point using the underlying environment dynamics. However, for many applications, particularly high-dimensional, real-world continuous control tasks, this transition manifold will generally be non-convex.

Neighborhood Mixup as a heuristic to encourage on-manifold interpolation. The aforementioned presence of non-convexity and nonlinearity in continuous control environments provides 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 domain "neighborhood", i.e. transitions with similar state-action pairs. This ensures that, provided the transition manifold is locally Euclidean (a property of manifolds), 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 MF, off-policy DRL algorithms.

Testing environment and configuration. We consider continuous control environments from the OpenAI Gym MuJoCo [3, 27] suite, namely: (i) `Ant-v2`, (ii) `HalfCheetah-v2`, (iii) `Hopper-v2`, (iv) `Swimmer-v2`, and (v) `Walker2d-v2`. Since we are principally interested in evaluating the sample efficiency of replay buffers, we treat the MF, off-policy DRL algorithms used for these evaluations (SAC [10] and TD3 [7]) as part of the experimental configuration. For each replay buffer variant, including NMER, we train agents using replay ratios (ratio of gradient steps: environment interactions) of 1, 5, and 20, and report the best results for each replay buffer variant. Additionally, for SAC [10], 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 Appendix.

Evaluation metric. We measure replay buffer sample efficiency using the evaluation reward of the reinforcement learning agent after 200K environment interactions have been sampled, as is performed in [16, 13]. Rewards are smoothed using an averaging window of 11, as is performed in [1].

Baselines. We compare NMER to the following baselines: (i) *Uniform, Vanilla Replay* (U) [18, 5], where transitions are sampled i.i.d. uniformly from the replay buffer, (ii) *Prioritized Experience Replay* (PER) [23] with stochastic prioritization, (iii) *Continuous Transition* (CT) [16]. 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 parameter 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 and the baselines we implement to SUNRISE [13] and other baselines tested using the same continuous control environments and 200K environment interaction evaluation [13]. Results for these baselines are provided in Table 3. Relative comparisons of NMER to additional baselines are in Table 2.

Learning curves for TD3 and SAC agents and NMER, U, PER, and CT baselines are depicted in Figures 3 and 4 (SUNRISE and other baselines evaluated in [13] are depicted with dashed horizontal lines indicating mean evaluation reward at 200K environment interactions), and comparative tabular results after 200K environment interactions are provided in Table 1. Each result is averaged over four runs. Note the following abbreviations in Table 1: (i) U = Uniform Replay, (ii) PER = Prioritized Experience Replay, (iii) CT = Continuous Transition, (iv) NMER = Neighborhood Mixup Experience Replay. Best results for each off-policy algorithm (TD3, SAC) are bolded.

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 [13].

Table 1: Continuous control experiment results (200K environment interactions)

RL Agent	Environment	U	PER	CT	NMER
TD3	Ant-v2	2005 ± 399	2317 ± 756	2834 ± 875	$\mathbf{4347 \pm 908}$
	HalfCheetah-v2	6467 ± 658	6447 ± 693	8097 ± 358	$\mathbf{9340 \pm 1678}$
	Hopper-v2	3252 ± 157	3213 ± 511	3156 ± 351	$\mathbf{3393 \pm 220}$
	Swimmer-v2	131 ± 20	$\mathbf{138 \pm 8}$	134 ± 10	122 ± 10
	Walker2d-v2	2236 ± 686	1452 ± 1057	3087 ± 1058	$\mathbf{4611 \pm 441}$
SAC	Ant-v2	1188 ± 692	800 ± 160	1594 ± 717	$\mathbf{2721 \pm 1685}$
	HalfCheetah-v2	4918 ± 1928	6880 ± 886	6120 ± 525	$\mathbf{8168 \pm 1585}$
	Hopper-v2	1692 ± 1160	$\mathbf{2801 \pm 827}$	1115 ± 897	1875 ± 900
	Swimmer-v2	110 ± 29	121 ± 42	106 ± 46	$\mathbf{140 \pm 10}$
	Walker2d-v2	4303 ± 636	3466 ± 784	$\mathbf{4696 \pm 1194}$	4429 ± 819

Table 2: Normalized comparative NMER performance of baselines evaluated on TD3.

Environment	U	PER	CT	NMER	NMER (K=1)	Mixup	S4RL	Noisy Replay
Ant-v2	0.461	0.533	0.652	**1.000**	0.449	0.543	0.177	0.393
HalfCheetah-v2	0.692	0.690	0.867	**1.000**	0.777	0.722	0.646	0.400
Hopper-v2	0.958	0.947	0.930	**1.000**	0.990	0.860	0.283	0.379
Swimmer-v2	1.074	**1.131**	1.098	1.000	0.910	0.352	0.361	0.320
Walker2d-v2	0.485	0.315	0.669	**1.000**	0.732	0.724	0.111	0.112
Average	0.734	0.723	0.843	**1.000**	0.771	0.640	0.316	0.321
Δ NMER (%)	-26.58%	-27.67%	-15.66%	**0.00%**	-22.85%	-35.97%	-68.44%	-67.93%

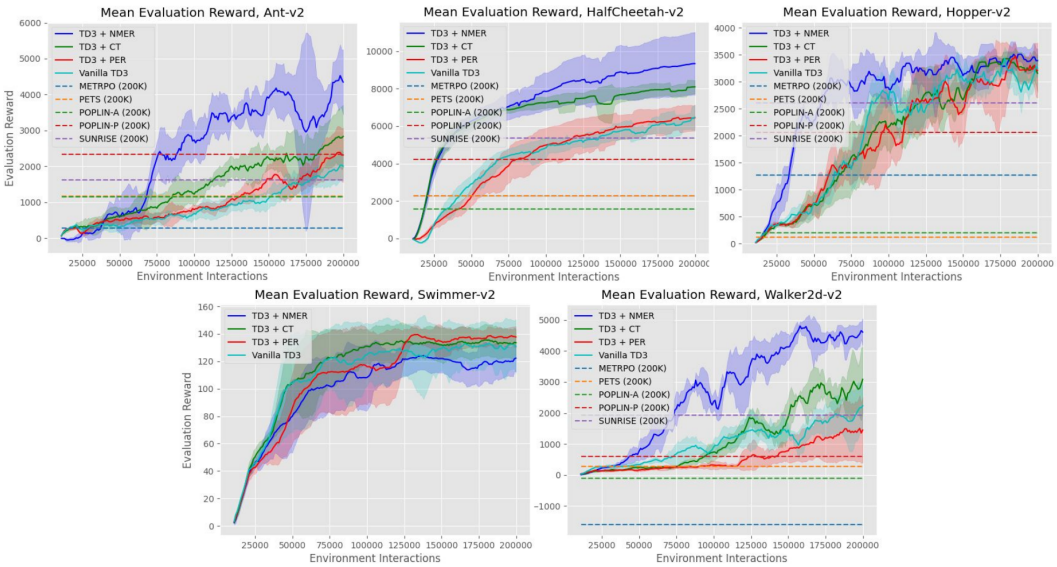

Figure 3: Learning curves for TD3 agents trained on NMER and baseline replay buffers. The blue learning curves correspond to agents trained with NMER. Rewards are reported as smoothed using an averaging window of 11 epochs, as is performed for the benchmarks in [1].

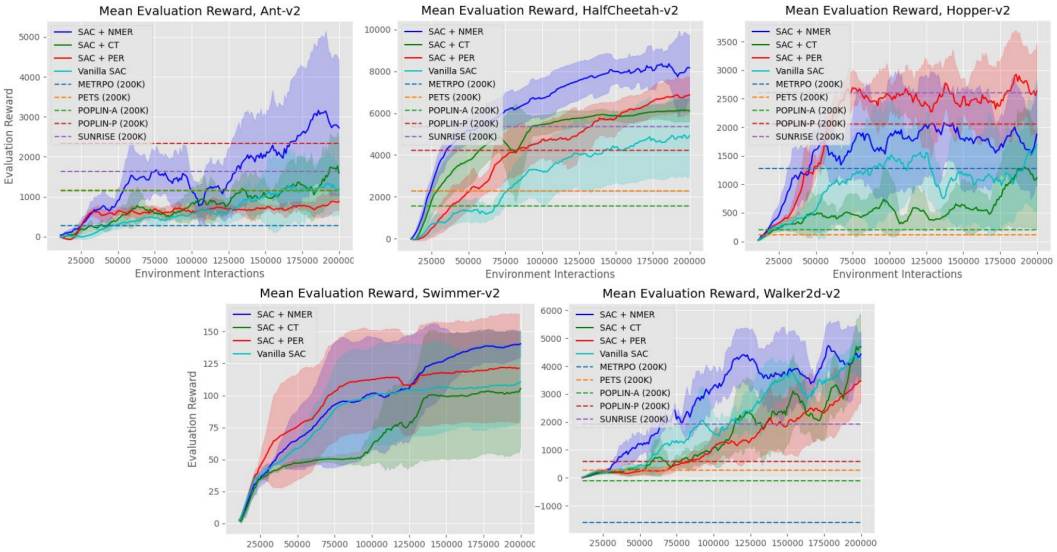

Figure 4: Learning curves for SAC agents trained on NMER and baseline replay buffers. The blue learning curves correspond to agents trained with NMER. Rewards are reported as smoothed using an averaging window of 11 epochs, as is performed for the benchmarks in [1].

Table 3: Baseline results from SUNRISE [13] (200K environment interactions)

Environment	METRPO	PETS	POPLIN-A	POPLIN-P	SUNRISE
Ant-v2	282 ± 18	1166 ± 227	1148 ± 438	2330 ± 321	1627 ± 293
HalfCheetah-v2	2284 ± 900	2288 ± 1019	1563 ± 1137	4235 ± 1133	5371 ± 483
Hopper-v2	1273 ± 501	115 ± 621	203 ± 963	2055 ± 614	2602 ± 307
Swimmer-v2	-	-	-	-	-
Walker2d-v2	-1609 ± 658	283 ± 502	-105 ± 250	597 ± 479	1926 ± 695

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 observed empirical success, NMER exhibits several limitations. NMER assumes that 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.

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 and 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. We empirically observe that local linear interpolation results in interpolated samples being close enough to the transition manifold to improve the sample efficiency of MF, off-policy DRL agents in the continuous control environments studied above. However, under some dynamics regimes and continuous control environments, local linear interpolation via local Mixup may result in the interpolated samples being significantly off-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 [19, 21].

7 Broader impact statement

Potential negative impacts. The use of synthetically-generated training transitions, even if these generated transitions are linear combinations of observed environment interactions, may reduce the interpretability of reinforcement learning agents trained with NMER, which could render safety and quality control assessments and guarantees for systems leveraging NMER, such as autonomous vehicles, more difficult to administer and/or prove. **Potential positive impacts.** Sample efficiency is a pivotal algorithmic property needed to practically apply DRL to real-world continuous control applications, such as autonomous vehicles. With sample-efficient experience replay approaches such as NMER, real-world reinforcement learning can be made more attainable, accessible, and robust, accelerating the integration of DRL to systems we interact with in our everyday lives.

8 Conclusion

We propose Neighborhood Mixup Experience Replay (NMER), an experience replay module for improving the sample efficiency of off-policy DRL agents. We demonstrate that by training agents on an infinite set of convex interpolated experiences generated via local Mixup in the transition space of a replay buffer, these agents can learn robust policies using fewer environment interactions compared to training solely on their observed experiences alone. NMER combines the benefits of learning from locally linear approximations of the underlying environment model with the sample efficiency benefits of model-based reinforcement learning methods, thus expanding the possibilities for tractable DRL in real-world continuous control settings.

Acknowledgements: 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 for providing HPC and consultation resources that have contributed to the research results reported within this publication.

References

[1] J. Achiam. Spinning up in deep reinforcement learning. 2018. 7, 8

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017. 1

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016. 3, 6

[4] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012. 2

[5] Y. Engel, S. Mannor, and R. 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. URL https://doi.org/10.1145/1102351.1102377. 7

[6] R. C. Fong, W. J. Scheirer, and D. D. Cox. Using human brain activity to guide machine learning. *Scientific reports*, 8(1):1–10, 2018. 1

[7] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018. 3, 6

[8] S. Gu, E. Holly, T. Lillicrap, and S. 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. 1

[9] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. 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. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html. 2

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy and A. 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. URL http://proceedings.mlr.press/v80/haarnoja18b.html. 3, 6

[11] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS. 2

[12] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. 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. URL https://proceedings.neurips.cc/paper/2020/file/e615c82aba461681ade82da2da38004a-Paper.pdf. 3

[13] K. Lee, M. Laskin, A. Srinivas, and P. Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. *arXiv preprint arXiv:2007.04938*, 2020. 7, 8

[14] S. Y. Lee, C. Sungik, and S.-Y. Chung. Sample-efficient deep reinforcement learning via episodic backward update. In *Advances in Neural Information Processing Systems*, pages 2112–2121, 2019. 1

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In Y. Bengio and Y. LeCun, editors, *ICLR*, 2016. 3

[16] J. Lin, Z. Huang, K. Wang, X. Liang, W. Chen, and L. Lin. Continuous transition: Improving sample efficiency for continuous control problems via mixup. *arXiv preprint arXiv:2011.14487*, 2020. 3, 7

[17] D. J. Mankowitz, G. Dulac-Arnold, and T. Hester. Challenges of real-world reinforcement learning. 2019. 1, 2

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 1, 2, 4, 7

[19] C. E. Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003. 9

[20] C. E. Rasmussen and Z. Ghahramani. Occam's razor. 2001. 4, 5

[21] C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems*, pages 751–759, 2004. 9

[22] E. Renaudo, B. Girard, R. Chatila, and M. 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. URL https://www.sciencedirect.com/science/article/pii/S1877050915036558. 6th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2015, 6-8 November Lyon, France. 2

[23] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *ICLR (Poster)*, 2016. 2, 3, 7

[24] W. Schwarting, T. Seyde, I. Gilitschenski, L. Liebenwein, R. Sander, S. Karaman, and D. Rus. Deep latent competition: Learning to race using visual control policies in latent space. *arXiv preprint arXiv:2102.09812*, 2021. 2

[25] T. Seyde, W. Schwarting, S. Karaman, and D. Rus. Learning to plan via deep optimistic value exploration. In A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. 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. URL http://proceedings.mlr.press/v120/seyde20a.html. 2

[26] S. Sinha and A. Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning. *arXiv preprint arXiv:2103.06326*, 2021. 3

[27] E. Todorov, T. Erez, and Y. 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. 3, 6

[28] W. B. P. von Pilchau, A. Stein, and J. Hähner. Bootstrapping a dqn replay memory with synthetic experiences. *arXiv preprint arXiv:2002.01370*, 2020. 3

[29] K. Wang, B. Kang, J. Shao, and J. Feng. Improving generalization in reinforcement learning with mixture regularization. *arXiv preprint arXiv:2010.10814*, 2020. 3

[30] T. Weyand, I. Kostrikov, and J. Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016. 2

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

[32] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani. Data efficient reinforcement learning for legged robots. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1–10. PMLR, 11 2020. URL http://proceedings.mlr.press/v100/yang20a.html. 1, 2

[33] Y. 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. URL https://doi.org/10.24963/ijcai.2018/820. 1

[34] D. Zha, K.-H. Lai, K. Zhou, and X. 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. URL https://doi.org/10.24963/ijcai.2019/589. 3

[35] H. Zhang, M. Cissé, Y. N. Dauphin, and D. 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. URL https://openreview.net/forum?id=r1Ddp1-Rb. 2, 3, 4, 5