# CRoSS: A Continual Robotic Simulation Suite for Scalable Reinforcement Learning with High Task Diversity and Realistic Physics Simulation

1st Yannick Denker
*Applied Computer Science*
*Fulda University of Applied Sciences*
Leipzigerstr. 123, 36037 Fulda, Germany
yannick.denker@cs.hs-fulda.de

2nd Alexander Gepperth
*Applied Computer Science*
*Fulda University of Applied Sciences*
Leipzigerstr. 123, 36037 Fulda, Germany
alexander.gepperth@cs.hs-fulda.de

***Abstract*—Continual reinforcement learning (CRL) requires agents to learn from a sequence of tasks without forgetting previously acquired policies. In this work, we introduce a novel benchmark suite for CRL based on realistically simulated robots in the Gazebo simulator. Our Continual Robotic Simulation Suite (CRoSS) benchmarks rely on two robotic platforms: a two-wheeled differential-drive robot with lidar, camera and bumper sensor, and a robotic arm with seven joints. The former represent an agent in line-following and object-pushing scenarios, where variation of visual and structural parameters yields a large number of distinct tasks, whereas the latter is used in two goal-reaching scenarios with high-level cartesian hand position control (modeled after the Continual World benchmark), and low-level control based on joint angles. For the robotic arm benchmarks, we provide additional kinematics-only variants that bypass the need for physical simulation (as long as no sensor readings are required), and which can be run two orders of magnitude faster. CRoSS is designed to be easily extensible and enables controlled studies of continual reinforcement learning in robotic settings with high physical realism, and in particular allow the use of almost arbitrary simulated sensors. To ensure reproducibility and ease of use, we provide a containerized setup (Apptainer) that runs out-of-the-box, and report performances of standard RL algorithms, including Deep Q-Networks (DQN) and policy gradient methods. This highlights the suitability as a scalable and reproducible benchmark for CRL research.**

***Index Terms*—Benchmarking, Continual Learning, Reinforcement Learning.**

## I. Introduction

Continual reinforcement learning (CRL) extends the classical reinforcement learning (RL) framework to settings where an agent must learn a sequence of tasks over time without access to previous environments. In contrast to standard RL, where the environment is typically assumed to be stationary and training data can be revisited indefinitely, CRL requires agents to continuously integrate new knowledge while preserving performance on previously encountered tasks. This continual setting is relevant for real-world applications such as robotics, autonomous driving, or online decision-making systems, where environmental conditions change over time and retraining from scratch is infeasible.

Author1 acknowledges support by Institution3

A core difficulty in CRL is catastrophic forgetting [1]–[4], where newly acquired knowledge interferes with previously learned policies. Although this problem is similar to the one faced in Continual Learning (CL), which deals with learning from non-stationary distributions, but without environment interaction, many established CL methods cannot easily be generalized to CRL. In particular, the assumption that each task is composed of well-defined sample *classes*, and that task onsets and classes are known to the learner, is what distinguishes CL from CRL.

### A. Motivation

Current CRL research commonly relies on benchmarks such as robotic reaching suites (e.g., Continual World [5]) and sequential game-based environments (e.g., Atari-based continual learning setups [6], [7]). While these benchmarks have been valuable for early progress, they exhibit several limitations that constrain their applicability. Robotic CRL benchmarks typically offer only a small number of distinct tasks, with limited support for sensors, plugins or actuator extensions. Many of these environments also do not work ”out of the box”, requiring substantial installation effort and relying on simulation backends that are difficult to configure, distribute, or extend. For Continual World, observations are low-dimensional, and the tasks do not really require learning in the first place, since the robot arm could be controlled directly using inverse kinematics. Additionally, the simulation is kinematics-only, containing no real physics engine. Game-based continual RL setups, including Atari, provide high-dimensional observations and actions with a wide variety of tasks. However, they operate in noise-free, fully deterministic environments that lack physical realism. Tasks are very diverse, but their intrinsic difficulty is very high, making it difficult to separate intrinsic difficulty from the difficulty of continual learning. We thus require a benchmark suite that incorporates out-of-the-box readiness, physical realism, a well-supported and extensible simulation backend, and a large number of tasks with high diversity.

Our goal is to close this gap by introducing the Continual Robotic Simulation Suite (**CRoSS**), a Gazebo-based CRL benchmark suite that combines realistic robotic simulation with massive task diversity while providing a reproducible, containerized deployment. CRoSS is designed around two complementary robotic platforms: First of all, a two-wheeled differential-drive robot is used in *multi-task line-following* (MLF) and *multi-task pushing-objects* (MPO) scenarios, where visual and structural parameters (e.g., line shape, textures, colors) are systematically varied to produce hundreds of distinct (but relatively simple) tasks. On the other hand, a 7-d.o.f. robotic arm is evaluated in *high-level reaching* (Cartesian end-effector control in six discrete directions mirroring the setup of Continual World) and *low-level reaching*, where the agent directly manipulates the seven arm joints, increasing task dimensionality and realism.

Our benchmark design leverages the Gazebo-Transport middleware communication between sensors, actuators, and agents. Since Gazebo and the Robot Operating System (ROS) are compatible through a standardized bridge, this control interface can be directly applied to physical robots with minimal modification, supporting extensions toward sim-to-real continual learning. Our environments can be used as drop-in replacements for standard Gymnasium tasks. The environment manager follows the same API conventions as Gymnasium making the benchmark directly compatible with existing RL pipelines and libraries. Through containerization, installation is simplified across a wide range of platforms.

### B. Contribution

We introduce a new benchmark suite for CRL with the following key contributions:

**Realistic robotic environments:** We present two Gazebo-based robotic platforms, a 2-wheeled differential-drive robot and a 7-degree-of-freeedom (d.o.f.) robot arm, enabling embodied CRL experiments in physically consistent simulations.
**High task diversity and scalability:** By varying visual and structural parameters, our benchmark generates hundreds of distinct tasks, allowing controlled studies of forgetting, transfer, and scalability in continual learning.
**Multiple control modalities:** The robotic arm is controlled both in cartesian and joint-space mode, allowing varying action complexities.
**Standardized, reproducible and extensible setup:** All environments are provided in a containerized (Apptainer) format that runs out of the box on any Linux system, using a well-supported simulation back-end that can be extended by a variety of community-developed plugins.
**Baseline evaluations and metrics:** We benchmark standard CRL algorithms, including DQN and REINFORCE, and provide evaluation metrics for measuring forgetting, transfer, and final performance.
**Simulation-to-real compatibility:** The benchmark's communication framework is built on Gazebo-Transport, which is inter-operable with ROS via an existing bridge, making it straightforward to port trained agents or environments from simulation to hardware platforms.
Together, these contributions establish a scalable, realistic, and easily extensible foundation for advancing continual reinforcement learning research in robotic domains. All benchmark environments, experiment scripts, and documentation are available in our public repository[1].

## II. Related Work

Comprehensive surveys have outlined the current state and challenges of the CRL field [8]–[10]. These works summarize existing methods and highlight the open problems in scalability, stability, and evaluation that motivate continued research in realistic and dynamic environments. More recently, generative replay and task-agnostic methods have emerged that adapt without explicit task boundaries (e.g., [11]–[13]) While these advances have improved algorithmic robustness, evaluation often remains confined to synthetic or low-dimensional environments that do not reflect the challenges of embodied learning.

Benchmark design has played a critical role in the development of CL algorithms. A recent survey of continual RL benchmarks [14] provides a broad overview of existing CRL benchmarks. Classical benchmarks such as Split-MNIST, Permuted-MNIST, or CIFAR-100 task sequences [15]–[19] primarily target supervised continual learning.

Game-based CRL benchmarks [7], [20]–[27] (e.g., Atari-derived or procedurally generated environments [28]–[30]) offer high visual diversity and scalability, but lack physical embodiment and robotics-specific constraints. CRoSS complements these by providing physically grounded environments where sensing, actuation, and embodiment are first-class components.

Physics-based benchmarks like Meta-World, Unity RL Playground, RLBench and robosuite [31]–[34] focus on manipulation skills in episodic or multi-task settings, typically with continuous control and limited task sequences. For other physics based benchmarks see [5], [35]–[39]. Continual World [5] extends Meta-World [31] to a continual setting, but remains restricted to kinematic control, low-dimensional observations, and a small number of tasks.

In contrast, CRoSS is explicitly designed around three orthogonal axes that are rarely combined: long task sequences with controlled overlap, enabling large-scale forgetting and transfer studies, heterogeneous robotic platforms, covering both mobile robotics and manipulation, and realistic robotic middleware and sensor integration via Gazebo and ROS. Rather than maximizing per-task difficulty, CRoSS emphasizes task *multiplicity*, allowing controlled variation of visual, structural, and task-semantic parameters while keeping individual tasks solvable in isolation.

Frameworks such as MuJoCo, PyBullet, and Gazebo [40]–[43] have enabled scalable robotic experiments, though most have been used for isolated single-task RL.

[1] https://github.com/anon-scientist/continual-robotic-simulation-suite/

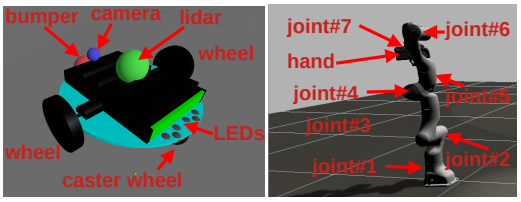


Fig. 1. Simulated robots. Left: two-wheeled $3\pi$ robot. Lidar sensors and camera objects are just visual placeholders to indicate their positions. Right: 7-d.o.f. Franka Emika Panda arm.

## III. Simulated Robots

### A. Two-wheeled mobile robot

This is a two-wheeled, differential-drive robot modeled after the Pololu 3Pi, see fig. 1 for a visual impression. The wheel separation is 10cm, and the wheel radii are 0.5cm each. It is equipped with a contact (bumper) sensor of radius 1cm placed in front, a lidar sensor placed in the center of the robot at a height over the ground of 4cm, and an RGB camera in various configurations. Although the position and orientation can be obtained from the simulator, we only use sensor data that could be obtained from a real robot as well. The lidar sensor covers a forward-looking angle range of $2\pi$ using 500 beams at a frequency of 300 Hz. Camera sensors can be configured to look down or ahead with a variable resolution. In addition to the differential-drive effector, the robot is equipped with 6 LEDs (red, blue, green, magenta, yellow, white) that can be controlled individually.

### B. 7-degree-of-freedom robot arm

This scenario employs a simulated, 7-d.o.f. robotic arm modeled after the Franka Emika Panda, see fig. 1 for a visual impression. The arm consists of seven joints providing a flexible kinematic structure that enables dexterous positioning of the end-effector in three-dimensional space. Each joint is actuated independently and reports its angular position and velocity, allowing precise low-level control. Ground-truth information, such as the end-effector pose, can be retrieved from the simulator for reward generation and policy evaluation.

## IV. Benchmarks

We provide four benchmark families spanning mobile robotics and manipulation. All benchmarks are designed such that individual tasks are solvable in isolation, while long task sequences induce catastrophic forgetting and negative transfer.

### A. Mobile Robot Benchmarks

We consider the two-wheeled differential-drive robot equipped with camera and lidar sensors described in section III-A. Two task families are defined: *multi-task line following (MLF)* and *multi-task pushing objects (MPO)*, illustrated in fig. 2. In **MLF**, the robot follows colored ground tracks composed of three parallel lines. Tasks are generated by varying line color combinations, yielding 150 distinct tracks. Each task consists of a small subset of tracks, with only one track replaced between successive tasks, inducing strong task overlap and gradual distribution shift. In addition to line following, the robot must activate task-specific LED signals at different track segments, coupling perception, control, and task identity. The choice of 6 LEDs to switch on, and three movement directions (left, right, straight) yields a discrete action space with 18 elements.

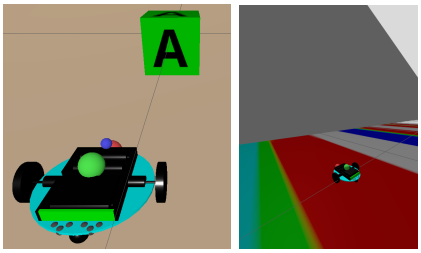


Fig. 2. Benchmarks for the two-wheeled robot used in this study. Left: multi-task pushing objects (MPO), the robot is approaching geometrical objects with different shapes, colors, and symbols projected onto their faces. Right: multi-task line following (MLF), the robot drives on a ground plane covered with colored lines and limited by a wall. It is supposed to follow the slim centered lines. The robot is the same in both benchmarks but uses a downwards-looking line camera in the MLF benchmark, whereas a forward-looking RGB camera is used in the MPO benchmark.

In **MPO**, the robot approaches geometric objects characterized by shape, color, and symbol. Whether contacting an object yields reward or punishment depends on its visual attributes. Tasks are again defined as small sets of object configurations with incremental replacement across tasks, resulting in long continual task sequences with shared structure. Here, the discrete action space contains the actions stop, straight, left and right.

Both MLF and MPO support three settings of increasing abstraction: a *default setting* (DS) using raw visual observations, a *simplified setting* (SS) using compact population-coded features, and a *super-simplified setting* (SSS) where parts of perception or control are handled by non-adaptive controllers. These variants allow disentangling task difficulty from continual learning difficulty while preserving identical task sequences. Reward definitions are deferred to the appendix.

### B. Reaching Benchmarks

We further provide reaching benchmarks based on a simulated 7-d.o.f. Franka Emika Panda arm described in section III-B.

In the **High-Level Reaching (HLR)** benchmark, tasks are defined by 3D target positions for the end-effector, following the structure of Continual World. The agent controls the arm via six discrete Cartesian movement actions. Observations include the current and goal end-effector positions. Inverse kinematics from PyKDL[2] is used to map Cartesian commands to joint configurations, with penalties applied for infeasible motions.

In the **Low-Level Reaching (LLR)** benchmark, the agent directly controls joint angles in a sequential manner while receiving only the current joint angles and the cartesian

[2]https://docs.ros.org/en/diamondback/api/kdl/html/python/

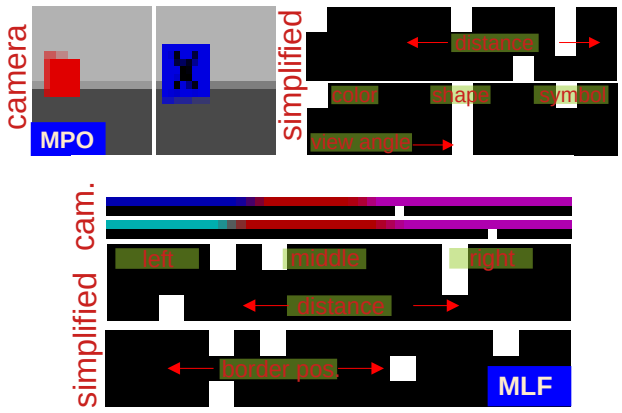


Fig. 3. Sensor data used for learning in the MLF and MPO benchmarks. Top: two default-setting 20x20 RGB images and two simplified 15x3 mono images from the MPO benchmark. Simplified inputs are population-coded, meaning that the position of the single bright pixel indicates a sensor reading. The uppermost row is split into three population codes for color, shape and symbol of the currently approached object, whereas the next rows encode object distance and viewing angle. Bottom: Two default-setting 100x2 composite images and two simplified 18x3 images from the MLF benchmark. Composite images integrate line camera data (upper row) and lidar distance to wall (lower row), the latter again population-coded. Simplified images contain three population codes for left/middle/right line color (upper row), distance to wall (middle row) and centering of the left border of the middle line in the image (population-coded). The simplified inputs are helpful because they simplify the problems without sacrificing their massively continual character.

target position as input. Each 7-step episode consists of one consecutive action per joint, requiring coordinated multi-joint control to reach a target position.

For both reaching benchmarks, we provide kinematics-only counterparts that replace physics simulation with analytical forward and inverse kinematics. These variants preserve task definitions, rewards, and termination criteria, but execute orders of magnitude faster, enabling large-scale experimentation and hyper-parameter studies. Key runtime comparisons are reported in the appendix.

## V. Experiments

Our experiments were conducted on a High-Performance Computing (HPC) cluster consisting of 40 Linux workstations, each equipped with an Nvidia GTX 3090 graphics card. We used TensorFlow 2.14 as the primary deep learning framework to implement and train our reinforcement learning models. All experiments were deployed using Apptainer containers, providing a reproducible environment.

### A. Multi-Task Line-Following (MLF)

Experiments are conducted using vanilla DQN for each setting: default (DS), simplified (SS) and super-simplified (SSS), see section IV-A. The benchmark consists of 150 tasks, which are executed one after the other for 300 (100 for SS and SSS) episodes per task using 30 steps per episode. Epsilon-greedy exploration is used, decreasing $\epsilon$ from either 1.0 (first task) or 0.5 (all other tasks) to a minimal value of 0.2 at the end of a task. A DNN with three hidden layers of size 100 maps observations to actions, where observations depend on the chosen setting. A replay buffer size of 15000 is used for all experiments, roughly 5 tasks' worth of samples. For each task $t' < t$, results from 10 exploitation-only episodes are averaged after training on task $t$. Due to the large number of tasks, we report evaluation performance on task 1 after training on tasks 1, 5, 10, 50, 90, 130 and 150. As an elementary performance measure, we use the average cumulated score per (test) episode, normalized by episode length. All results are obtained by averaging three independent runs with identical parameters. The results are summarized in table II. In addition to the CRL experiments, we also investigated the intrinsic difficulty of individual tasks, randomly training on tasks 1–5 and 105–110 for this purpose. In table I, we observe that all tasks can be learned to a similar degree of precision.

TABLE I
Evaluation of randomly picked individual tasks in the MPO and MLF benchmarks for all settings. We observe a small degradation between SSS and DS, which may be explained by increased task difficulty. MPO results differ between tasks according to whether to target object should be pushed (+10 reward) or not pushed (only small reward for stopping in time).

| task→ | 51 | 52 | 53 | 54 | 106 | 107 | 108 | 109 |
|---|---|---|---|---|---|---|---|---|
| ↓ setting | MLF | | | | | | | |
| DS | 1.45 | 1.48 | 1.44 | 1.48 | 1.49 | 1.44 | 1.42 | 1.41 |
| SS | 1.52 | 1.52 | 1.51 | 1.49 | 1.54 | 1.51 | 1.52 | 1.50 |
| SSS | 1.61 | 1.58 | 1.61 | 1.61 | 1.57 | 1.62 | 1.61 | 1.61 |
| ↓ setting | MPO | | | | | | | |
| DS | 21.0 | 11.3 | 22.7 | 12.0 | 21.5 | 11.5 | 21.5 | 12.0 |
| SS | 21.3 | 11.8 | 20.9 | 12.9 | 22.0 | 11.8 | 21.8 | 11.9 |
| SSS | 23.5 | 13.2 | 23.9 | 13.9 | 23.0 | 12.5 | 23.6 | 13.7 |

### B. Multi-task Pushing-Objects (MPO)

Experiments are conducted using vanilla DQN for each setting: default (DS), simplified (SS) and super-simplified (SSS), see section IV-A. The benchmark consists of 125 tasks (each including four tracks), which are executed one after the other for 300 (100 for SS and SSS) episodes per task using 30 steps per episode. Epsilon-greedy exploration is used, decreasing $\epsilon$ from either 1.0 (first task) or 0.5 (all other tasks) to a minimal value of 0.2 at the end of a task. A DNN with three hidden layers of size 100 maps observations to actions, where observations depend on the chosen setting. A replay buffer size of 15000 is used for all experiments, roughly 5 tasks' worth of samples. For each task $t' < t$, results from 10 exploitation-only episodes are averaged after training on task $t$. Due to the large number of tasks, we report evaluation performance on task 1 after training on tasks 1, 5, 10, 50, 100 and 125. As an elementary performance measure, we use the average cumulated score per (test) episode. All results are obtained by averaging three independent runs with identical parameters. The results are summarized in table II. Run-time is approximately 15 minutes per task or approximately 30 hours in total.

In addition to the CRL experiments, we also investigated the intrinsic difficulty of individual tasks, randomly training on tasks 1–4 and 106–110 for all settings. In table I, we observe that all tasks where touching the object is expected reach

TABLE II

EXPERIMENTAL RESULTS FOR THE MPO AND MLF BENCHMARKS. SHOWN IS AVERAGE REWARD PER STEP FOR THE MLF BENCHMARK AND AVERAGE CUMULATED REWARD FOR THE MPO BENCHMARK, EVALUATED FOR TASK 1 AFTER HAVING TRAINED ON TASK $t$. THE RETENTION OF THE FIRST TASK IS A SIMPLE YET INFORMATIVE MEASURE THAT RETAINS MEANINGFULNESS EVEN WHEN THE NUMBER OF TASKS IS LARGE. CLEAR SIGNS OF FORGETTING CAN BE OBSERVED FOR ALL BENCHMARKS AND SETTINGS.

| Benchm. → | MLF | | | | | | MPO | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ↓ Setting | evaluated after task | | | | | | evaluated after task | | | | | |
| | 1 | 5 | 10 | 70 | 120 | 150 | 1 | 5 | 10 | 70 | 120 | 150 |
| DS | 1.41 | 1.31 | 1.05 | -0.21 | 0.10 | -0.54 | 11.8 | 11.9 | 3.0 | -2.0 | -1.1 | 4.0 |
| SS | 1.51 | 1.52 | 0.99 | 0.11 | -0.55 | 0.05 | 12.3 | 11.5 | 2.7 | 3.5 | -3.1 | -3.6 |
| SSS | 1.59 | 1.55 | 1.49 | 0.33 | -0.75 | -0.32 | 13.8 | 11.2 | 11.3 | 0.7 | -1.2 | -1.7 |

approximately the same reward, and the same holds for tasks where pushing the object is punished, although the reward is lower in the latter case because the robot only receives the reward for driving towards the object. We thus conclude that all tasks can be individually solved by the robot, and any performance degradation originates in the sequential nature of learning.

### C. High-Level Reaching (HLR)

For the HLR benchmark we conduct a series of experiments using a Deep Q-Network (DQN) as baseline learner. The overall task configuration is summarized in table III, comprising ten distinct reaching tasks that differ in their target goal positions within the 3D workspace.

Each experiment consists of a sequential training run across all ten tasks. For every task $T_i$, the agent is trained for 5000 steps while retaining the learned model parameters between tasks. After completing each task, the agent is evaluated on all previously encountered tasks ($T_0 ... T_i$) for 20 test episodes per task to measure retention and transfer effects. Each full experiment is repeated three times using identical hyperparameters. The baseline learner is a standard DQN implementation. The DQN agent is implemented in TensorFlow using an Adam optimizer with a learning rate of $1 * 10^-4$ and a discount factor $\gamma = 0.8$. The network consists of two fully connected layers with 128 and 64 hidden units, respectively. Training is performed with a batch size of 32, sampling uniformly from a replay buffer whose capacity was varied between 5000, 10000, and 20000 experiences to analyze the effect of memory size. Exploration follows an $\epsilon$-greedy strategy, starting from an initial $\epsilon = 1.0$ that linearly decayed by $\Delta_\epsilon = 0.0002$ until reaching a minimum of $\epsilon = 0.2$ At the end of each training phase, the current policy is evaluated without further learning updates.

Performance is measured using two complementary metrics: the **average step reward**, computed as the mean cumulative step reward over 20 evaluation episodes, and the **accuracy**, defined as the fraction of episodes in which the goal is reached within the success tolerance. After each task, both metrics are computed for all previously seen tasks yielding a continual evaluation matrix that captures both learning and forgetting over time. The results are aggregated over four independent runs to reduce stochastic variance and are reported in table IV. The final accuracies for all tasks reach approximately **11%**, **23%** and **39%** for buffer sizes of 5000, 10000 and 50000 at the end of training.

To confirm the learnability of individual tasks in isolation, each of the ten tasks is also trained independently (non-sequentially). Using the same DQN configuration with a replay buffer size of 200, all tasks converge to 100% accuracy across three runs, demonstrating that each task is individually solvable and that performance degradation in the sequential setup is due to catastrophic forgetting rather than task difficulty. Most parameter fine-tuning and preliminary experiments is conducted in the kinematic variant of the benchmark. This approach provides an identical control interface while offering a significant reduction in computational cost, training runs in the kinematic setup are several orders of magnitude faster than in the full Gazebo simulation (see appendix for a comparison). All final results are subsequently verified in the physical simulation to ensure that the observed learning behavior and performance trends remain consistent in realistic, dynamically simulated conditions.

### D. Low-Level Reaching (LLR)

The experimental procedure is analogous to that of the HLR benchmark, with a few key modifications to accommodate the joint-space formulation. Each task $T_i$ is trained for 20000 steps, and after completing training on task $T_i$, the agent is evaluated on all previously seen tasks once. All other hyperparameters, including network architecture, optimizer, learning rate, discount factor, batch size, and exploration parameters, remain identical to those used in the HLR benchmark and are described in detail in the that section.

Unlike in the HLR experiment, where the agent receives incremental feedback at every step, the low-level variant employs a REINFORCE-style policy gradient approach in which only the final reward at the end of each episode is of relevance. Since the true task outcome, the closeness of the end-effector to the goal, can only be determined after all seven joints have been moved, it is back-propagated through the entire episode using a discount factor of $\gamma = 1.0$, consistent with standard REINFORCE updates for episodic problems. Evaluation follows the same general methodology as the HLR benchmark but is adapted to the episodic reward structure. Performance is reported as the average final reward per step and success accuracy (fraction of episodes reaching the goal within a given tolerance). Results are averaged over 4 independent runs and summarized in table V.

Each of the eight low-level tasks is also trained independently using the same DQN configuration and a replay buffer size of 200. All tasks reach 100% accuracy across two

TABLE III
TASKS ARE MODELED AFTER THOSE INTRODUCED IN THE *Continual-World* BENCHMARK [5].

| Task | Name |
|---|---|
| T1 | hammer |
| T2 | push wall |
| T3 | faucet close |
| T4 | push back |
| T5 | stick pull |
| T6 | handle press |
| T7 | push ball |
| T8 | shelf place |
| T9 | window close |
| T10 | peg unplug |

TABLE IV
AVERAGE STEP REWARD FOR THE DQN BASELINE WITH A REPLAY BUFFER CAPACITY OF 5000 ON THE KINEMATIC HIGH-LEVEL REACHING (HLR-K) BENCHMARK. THE REPORTED VALUES REPRESENT AVERAGES OVER MULTIPLE EVALUATION EPISODES AND INDEPENDENT RUNS WITH IDENTICAL PARAMETERS

| Task | Evaluation after | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |
| T1 | 0.71 | 0.70 | 0.70 | 0.70 | 0.70 | 0.72 | 0.64 | 0.61 | 0.62 | 0.58 |
| T2 | - | 0.65 | -0.33 | -0.24 | 0.64 | 0.63 | 0.46 | 0.51 | 0.53 | 0.59 |
| T3 | - | - | 0.53 | 0.58 | 0.65 | 0.69 | -1.18 | -0.34 | -1.32 | 0.64 |
| T4 | - | - | - | 0.77 | 0.77 | 0.77 | 0.44 | 0.42 | 0.47 | 0.43 |
| T5 | - | - | - | - | 0.60 | 0.55 | 0.58 | 0.66 | 0.42 | 0.59 |
| T6 | - | - | - | - | - | 0.54 | 0.54 | 0.02 | 0.47 | 0.59 |
| T7 | - | - | - | - | - | - | 0.65 | 0.65 | 0.65 | 0.65 |
| T8 | - | - | - | - | - | - | - | 0.67 | 0.67 | 0.68 |
| T9 | - | - | - | - | - | - | - | - | 0.67 | 0.64 |
| T10 | - | - | - | - | - | - | - | - | - | 0.54 |

TABLE V
FINAL-STEP REWARDS FOR DQN WITH A REPLAY BUFFER OF 10000 ON THE KINEMATIC LOW-LEVEL REACHING (LLR-K) BENCHMARK. REPORTED VALUES ARE AVERAGES OVER MULTIPLE EVALUATION EPISODES AND INDEPENDENT RUNS WITH IDENTICAL PARAMETERS.

| Task | Evaluation after | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |
| T1 | 1.0 | 1.0 | 0.22 | -0.05 | -0.04 | -0.29 | -0.01 | 0.02 |
| T2 | - | 0.55 | 0.59 | 0.23 | 0.18 | 0.19 | 0.50 | 0.24 |
| T3 | - | - | 0.87 | 0.86 | 0.11 | 0.32 | -0.24 | 0.66 |
| T4 | - | - | - | 0.47 | 0.79 | 0.53 | 0.43 | 0.57 |
| T5 | - | - | - | - | 1.0 | 1.0 | 1.0 | 0.60 |
| T6 | - | - | - | - | - | 0.52 | 1.0 | 0.52 |
| T7 | - | - | - | - | - | - | 0.75 | 1.0 |
| T8 | - | - | - | - | - | - | - | 0.53 |

runs, confirming that they are individually solvable and that performance degradation in the sequential setup arises from catastrophic forgetting rather than inherent task difficulty. Just like HLR, hyper-parameter selection is performed using the kinematic environment variant to enable rapid experimentation, and all results are verified in the simulated Gazebo setup.

## VI. DISCUSSION

**Principal conclusions** The results of the previous section confirm that our benchmark captures essential aspects of continual reinforcement learning (CRL). Across all experiments, standard reinforcement learning methods such as DQN and REINFORCE show strong signs of catastrophic forgetting when trained sequentially on the proposed task sequences. While individual tasks are fully solvable, performance on earlier tasks consistently deteriorates as new ones are introduced, depending on the size of the replay buffer. On the other hand, replay buffer size cannot be unbounded, and even if it could, a larger replay buffer implies that learning of new policies is slowed down, see also [44].

**Choice of algorithms** The goal of this article is to describe the advantages of using the benchmark suite, not on particular CRL algorithms. We therefore demonstrate the performance of standard RL algorithms, like DQN and REINFORCE to establish a baseline. Other RL algorithms, like SAC or PPO, are difficult to adapt to the CRL setting since exploration can only be controlled implicitly, as was shown in [45], and are thus not studied here.

**Comparison to Continual World and Reaching Benchmarks** CRoSS deliberately builds on the conceptual simplicity of Continual World while extending it along several dimensions that are critical for continual reinforcement learning research. Our benchmarks separates control abstraction from task definition by providing both high-level Cartesian control and low-level joint-space control within the same benchmark suite. While Continual World exclusively relies on end-effector translations, the low-level reaching benchmark in CRoSS introduces sequential, joint-wise decision making, significantly increasing action dimensionality without changing the underlying task semantics. We explicitly decouples task difficulty from continual difficulty. All reaching, line-following, and object-pushing tasks are fully solvable in isolation, as demonstrated by single-task experiments. Performance degradation therefore arises from sequential interference rather than insufficient task learnability. This property is essential for studying catastrophic forgetting and transfer effects in long task sequences, and is not guaranteed in many existing benchmarks. Additionally CRoSS extends continual evaluation beyond manipulation by introducing mobile robot scenarios with rich sensing modalities. This enables the study of continual learning phenomena in navigation-like settings, where perception, control, and task identity are tightly coupled. Importantly, these scenarios are designed to allow systematic task overlap, ensuring gradual task drift rather than abrupt task replacement.

**Scalable Task Generation with Controlled Overlap** CRoSS is designed to generate large numbers of distinct tasks without handcrafted environments by composing a small set of interpretable factors (e.g., colors, shapes, layouts, or goal positions). Task sequences are constructed with controlled overlap, where only a subset of components changes between successive tasks, inducing gradual distribution shift rather than abrupt task replacement. This factorized design allows scalable continual learning evaluation, ensuring that forgetting and transfer arise from interference between related, individually solvable tasks rather than from unrelated task switches.

**Extensibility and stability** Key advantages of building on Gazebo are extensibility and long-term stability. New robotic setups, sensors, or task environments can be added with minimal effort, as Gazebo already provides a wide array of community-supported sensor types, including RGB and depth cameras, lidar and contact sensors, that can be directly

exposed as observation channels during training. Furthermore, the Gazebo simulator is well-supported by LTS versions for at least the next decade, offering a stable and standardized platform for further research. CRoSS' modular architecture enables straightforward expansion with new robots, tasks, and learning paradigms. We designed it to be easily integrated into existing Gymnasium-based workflows. Our environment manager mirrors the Gymnasium interface (e.g., `reset()`, `step()`, termination handling, and observation/action formatting), enabling users to train agents with Gymnasium-compatible algorithms without major modification. Because all scenarios are defined through Gazebo world files and parameterized configuration scripts, additional tasks can be easily implemented.

**ROS compatibility** Since the benchmark uses the Gazebo-Transport communication system, which is inter-operable with ROS through the official ROS-Gazebo bridge, trained policies can easily be transferred to and executed on real-world robots using identical message structures. This interoperability makes our benchmark not only a simulation platform but also a bridge between simulated and real robotic learning, allowing researchers to evaluate sim-to-real transfer in continual reinforcement learning contexts.

**Bonus: kinematics-only reaching** CRoSS provides kinematics-only counterparts for its reaching benchmarks. These variants preserve task definitions, reward functions, and control interfaces, while enabling orders-of-magnitude faster experimentation (see appendix for a comparison). This makes it feasible to perform large-scale ablations, hyper-parameter studies, and algorithmic comparisons, while still validating key findings in a physically simulated environment.

## VII. Future work

The presented benchmark suite could be extended in several ways. First of all, we could define a large number of additional tasks for the reaching benchmarks by replicating existing tasks but varying the initial posture of the arm. Then, the reaching benchmarks could be made more realistic by replacing the end-effector 3D position in the observations by, e.g., data from a camera or lidar sensor. Lastly, a comparison of current state-of-the-art CRL approaches could be conducted using the benchmark suite.

## References


[1] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[2] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[3] C. V. Nguyen, A. Achille, M. Lam, T. Hassner, V. Mahadevan, and S. Soatto, "Toward understanding catastrophic forgetting in continual learning," *arXiv preprint arXiv:1908.01091*, 2019.

[4] X. Li, Y. Zhou, T. Wu, R. Socher, and C. Xiong, "Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting," in *International conference on machine learning*. PMLR, 2019, pp. 3925–3934.

[5] M. Wolczyk, M. Zajac, R. Pascanu, L. Kucinski, and P. Milos, "Continual world: A robotic benchmark for continual reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 496–28 510, 2021.

[6] Z. Abbas, R. Zhao, J. Modayil, A. White, and M. C. Machado, "Loss of plasticity in continual deep reinforcement learning," in *Conference on lifelong learning agents*. PMLR, 2023, pp. 620–636.

[7] Q. Delfosse, J. Blüml, B. Gregori, and K. Kersting, "Hackatari: Atari learning environments for robust and continual reinforcement learning," *arXiv preprint arXiv:2406.03997*, 2024.

[8] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," *Journal of Artificial Intelligence Research*, vol. 75, pp. 1401–1476, 2022.

[9] H. Lyu, N. Sha, S. Qin, M. Yan, Y. Xie, and R. Wang, "Advances in neural information processing systems," *Advances in neural information processing systems*, vol. 32, 2019.

[10] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, "Embracing change: Continual learning in deep neural networks," *Trends in cognitive sciences*, vol. 24, no. 12, pp. 1028–1040, 2020.

[11] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Information fusion*, vol. 58, pp. 52–68, 2020.

[12] K. Khetarpal, M. Klissarov, M. Chevalier-Boisvert, P.-L. Bacon, and D. Precup, "Options of interest: Temporal abstraction with interest functions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4444–4451.

[13] M. Wołczyk, B. Wójcik, K. Bałazy, I. T. Podolak, J. Tabor, M. Śmieja, and T. Trzcinski, "Zero time waste: Recycling predictions in early exit neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2516–2528, 2021.

[14] C. Pan, X. Yang, Y. Li, W. Wei, T. Li, B. An, and J. Liang, "A survey of continual reinforcement learning," *arXiv preprint arXiv:2506.21872*, 2025.

[15] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[16] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[17] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[18] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International conference on machine learning*. PMLR, 2017, pp. 3987–3995.

[19] G. M. Van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*, 2019.

[20] S. Powers, E. Xing, E. Kolve, R. Mottaghi, and A. Gupta, "Cora: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents," in *Conference on Lifelong Learning Agents*. PMLR, 2022, pp. 705–743.

[21] T. Tomilin, M. Fang, Y. Zhang, and M. Pechenizkiy, "Coom: A game benchmark for continual reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 67 794–67 832, 2023.

[22] H. Küttler, N. Nardelli, A. Miller, R. Raileanu, M. Selvatici, E. Grefenstette, and T. Rocktäschel, "The nethack learning environment," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7671–7684, 2020.

[23] D. Kim, "Uncertainty-aware continual reinforcement learning via ppo with graph representation learning," *Mathematics*, vol. 13, no. 16, p. 2542, 2025.

[24] M. Samvelyan, R. Kirk, V. Kurin, J. Parker-Holder, M. Jiang, E. Hambro, F. Petroni, H. Küttler, E. Grefenstette, and T. Rocktäschel, "Minihack the planet: A sandbox for open-ended reinforcement learning research," *arXiv preprint arXiv:2109.13202*, 2021.

[25] E. C. Johnson, E. Q. Nguyen, B. Schreurs, C. S. Ewulum, C. Ashcraft, N. M. Fendley, M. M. Baker, A. New, and G. K. Vallabha, "L2explorer: A lifelong reinforcement learning assessment environment," *arXiv preprint arXiv:2203.07454*, 2022.

[26] H. Nekoei, A. Badrinaaraayanan, A. Courville, and S. Chandar, "Continuous coordination as a realistic scenario for lifelong learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8016–8024.

[27] V. Lomonaco, K. Desai, E. Culurciello, and D. Maltoni, "Continual reinforcement learning in 3d non-stationary environments," in *Proceed-*

*ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 248–249.
[28] C. Brennan, A. Williams, O. G. Younis, V. Vyas, D. Yasafova, and I. Rish, "Using unity to help solve reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 2484–2506, 2024.
[29] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *International conference on machine learning*. PMLR, 2020, pp. 2048–2056.
[30] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of artificial intelligence research*, vol. 47, pp. 253–279, 2013.
[31] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on robot learning*. PMLR, 2020, pp. 1094–1100.
[32] L. Ye, R. Li, X. Hu, J. Li, B. Xing, Y. Peng, and B. Liang, "Unity rl playground: A versatile reinforcement learning framework for mobile robots," *arXiv preprint arXiv:2503.05146*, 2025.
[33] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.
[34] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu, "robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2020.
[35] F. Yang, C. Yang, H. Liu, and F. Sun, "Evaluations of the gap between supervised and reinforcement lifelong learning on robotic manipulation tasks," in *Conference on Robot Learning*. PMLR, 2022, pp. 547–556.
[36] G. I. Parisi and V. Lomonaco, "Online continual learning on sequences," in *Recent Trends in Learning From Data: Tutorials from the INNS Big Data and Deep Learning Conference (INNSBDDL2019)*. Springer, 2020, pp. 197–221.
[37] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
[38] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska, "Urban driver: Learning to drive from real-world demonstrations using policy gradients," in *Conference on Robot Learning*. PMLR, 2022, pp. 718–728.
[39] N. Lucchesi, A. Carta, V. Lomonaco, and D. Bacciu, "Avalanche rl: A continual reinforcement learning library," in *International Conference on Image Analysis and Processing*. Springer, 2022, pp. 524–535.
[40] 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*. IEEE, 2012, pp. 5026–5033.
[41] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.
[42] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. Ieee, 2004, pp. 2149–2154.
[43] D. Ferigo, S. Traversaro, G. Metta, and D. Pucci, "Gym-ignition: Reproducible robotic simulations for reinforcement learning," in *2020 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2020, pp. 885–890.
[44] S. Zhang and R. S. Sutton, "A deeper look at experience replay," *arXiv preprint arXiv:1712.01275*, 2017.
[45] P. Osinenko, G. Yaremenko, D. Belov, and A. Gepperth, "Adiabatic reinforcement learning," in *Internation Conference on Advanced robotics (ICAR)*, 2025.

## APPENDIX

**Reproducibility and Additional Details** All benchmark code, configuration files, and instructions are available in our (currently anonymized) repository: https://github.com/anon-scientist/continual-robotic-simulation-suite/. Additional benchmark details (reward definitions, full observation/action specifications, and extended results) are provided there.

**Runtime reduction through kinematic benchmark variants** To quantify the computational benefits of the analytical benchmark variants, we compare the average total experiment completion time for both the physically simulated and kinematic versions of the High-Level Reaching (HLR) and Low-Level Reaching (LLR) benchmarks over multiple runs with the same parameter setup. The results are summarized in table VI.

TABLE VI
AVERAGE TOTAL TRAINING AND EVALUATION TIME MEASURED FOR THE HIGH-LEVEL REACHING BENCHMARKS.

| | Simulated | Kinematic |
|---|---|---|
| **HLR** | ≈33h 30min | ≈ 42min |
| **LLR** | ≈10h 55min | ≈ 40min |

The performance difference arises primarily from the physical simulation overhead inherent to Gazebo. Even though the simulation was run at 10x real-time speed, faster execution is impractical because the physics simulation becomes a bottleneck. Each simulation step incurs a non-negligible delay from physics updates, sensor readings, and inter-process communication.

**Reward computations**

*(a) Multi-task Line Following (MLF)*

$$r_t = r_t^{\text{lf}} + r_t^{\text{LED}}, \qquad r_t^{\text{LED}} \in \{+1, -1\},$$
$$r_t^{\text{lf}} = 1 - \left| \frac{d_t - \frac{W}{2}}{\frac{W}{2}} \right|, \qquad d_t \in [0, W]. \tag{1}$$

Here $d_t$ is the detected left line-edge pixel and $W{=}100$. If the line is lost, terminate with $r_t^{\text{lf}} = -1$.

*(b) Multi-task Pushing Objects (MPO)*

$$r_t = r_t^a + r_t^p,$$
$$r_t^p = \begin{cases} +10, & \text{pushable touched} \\ -10, & \text{non-pushable touched} \\ 0, & \text{otherwise} \end{cases}$$
$$r_t^a(\vec{o}_t, a_t) = (1 - |\phi_t/25|) \times r_t^s$$
$$r_t^{s(a_t)} = \begin{cases} 0.1, & a_t = \text{stop} \\ 1, & \text{else} \end{cases} \tag{2}$$

$\phi_t$ is the lidar-based viewing angle (deg). Touch terminates; if $r_t^a < 0$, terminate with reward $-1$.

*(c) Low-level Reaching (LLR)*

$$r = 1 - \left\| \vec{p}_{\text{final}} - \vec{p}_{\text{goal}} \right\|_2. \tag{3}$$

*(d) High-level Reaching (HLR)*

$$r_t = \begin{cases} 1, & \|\vec{p}_{\text{curr}} - \vec{p}_{\text{goal}}\|_2 < 0.1, \\ 1 - \|\vec{p}_{\text{curr}} - \vec{p}_{\text{goal}}\|_2, & \text{otherwise.} \end{cases} \tag{4}$$

Episode ends on success or after 30 steps; infeasible moves yield a small negative reward and no state change.