Real Robot Challenge 2022: Learning Dexterous Manipulation from Offline Data in the Real World

Nico Gürtler[1], **Felix Widmaier**[1], **Cansu Sancaktar**[1], **Sebastian Blaes**[1], **Pavel Kolev**[1], **Stefan Bauer**[2], **Manuel Wüthrich**[3], **Markus Wulfmeier**[4], **Martin Riedmiller**[4], **Arthur Allshire**[5], **Qiang Wang**[†6], **Robert McCarthy**[†6], **Hangyeol Kim**[†7], **Jongchan Baek**[†7], **Wookyong Kwon**[†8], **Shanliang Qian**[†], **Yasunori Toshimitsu**[†9], **Mike Yan Michelis**[†9], **Amirhossein Kazemipour**[†9], **Arman Raayatsanati**[†9], **Hehui Zheng**[†9], **Barnabasa Gavin Cangan**[†9], **Bernhard Schölkopf**[1], and **Georg Martius**[1]

[1] *Max Planck Institute for Intelligent Systems, Tübingen, Germany*

[2] *Helmholtz and TU Munich*

[3] *Harvard University*

[4] *DeepMind*

[5] *University of Toronto*

[6] *University College Dublin*

[7] *Pohang University of Science and Technology*

[8] *Electronics and Telecommunications Research Institute*

[9] *ETH Zurich*

[†] *Competition participants*

Editors: Marco Ciccone, Gustavo Stolovitzky, Jacob Albrecht

Abstract

Experimentation on real robots is demanding in terms of time and costs. For this reason, a large part of the reinforcement learning (RL) community uses simulators to develop and benchmark algorithms. However, insights gained in simulation do not necessarily translate to real robots, in particular for tasks involving complex interactions with the environment. The *Real Robot Challenge 2022*[1] therefore served as a bridge between the RL and robotics communities by allowing participants to experiment remotely with a *real* robot – as easily as in simulation.

In the last years, offline reinforcement learning has matured into a promising paradigm for learning from pre-collected datasets, alleviating the reliance on expensive online interactions. We therefore asked the participants to learn two dexterous manipulation tasks involving pushing, grasping, and in-hand orientation from provided real-robot datasets. An extensive software documentation and an initial stage based on a simulation of the real set-up made the competition particularly accessible. By giving each team plenty of access budget to evaluate their offline-learned policies on a cluster of seven identical real TriFinger platforms, we organized an exciting competition for machine learners and roboticists alike.

In this work we state the rules of the competition, present the methods used by the winning teams and compare their results with a benchmark of state-of-the-art offline RL algorithms on the challenge datasets.

Keywords: Reinforcement Learning, Robotics, Manipulation, Competition, Offline RL

1. https://real-robot-challenge.com/

1. Introduction

Robots have the potential to help humans in many tasks provided that they are adaptive and versatile. Learning methods are a promising route to creating such flexible control strategies, as they can learn to cope with the complexities of the real world. Indeed, reinforcement learning (RL) approaches have recently achieved good performance in challenging robotics tasks (Kalashnikov et al., 2018; OpenAI et al., 2019; Rudin et al., 2022). However, training such policies requires either a large number of expensive and potentially unsafe environment interactions (Dulac-Arnold et al., 2020) or a good simulator. The field of offline RL (Lange et al., 2012; Levine et al., 2020; Prudencio et al., 2022) therefore aims to learn from pre-existing datasets without the need for online interactions.

This paradigm could potentially have a transformative effect on robotics similar to the impact of large datasets in supervised learning. Yet, as the experiments on real robots are costly and time consuming, the offline RL community mostly benchmarks their algorithms in simulated environments (Fu et al., 2020). It is, however, not clear to which extent results obtained in simulation transfer to the real world with its noisy and delayed observations and complex dynamics.

To fill this gap, we organized the *Real Robot Challenge 2022* which was hosted at NeurIPS 2022. We asked the community to learn two dexterous manipulation tasks from pre-collected real-robot datasets we provided, using either offline RL or imitation learning. We chose dexterous manipulation as a challenge as it is a fundamental building block for more complex tasks and a challenging research topic in its own right. Participants could evaluate their solutions remotely by submitting them to a cluster of TriFinger robots (Wüthrich et al., 2020) hosted at the Max Planck Institute for Intelligent Systems, Tübingen, Germany.

In the rest of this paper, we describe the challenge in detail in section 3, explain the data collection in 4, present the baselines and top submissions in section 5 and discuss the results in section 6. Finally, we summarize takeaways from the competition in section 6.3.

2. Related Work

There have been numerous reinforcement learning competitions for continuous control problems at top machine learning conferences. However, almost all of them exclusively focused on scenarios in simulation. For instance, in the NeurIPS competition track from 2019[2] and 2020[3], the only competition involving real robots was AI Driving Olympics 3 and 5, which provided a toy environment aiming to replicate a real system via miniaturized self-driving cars (Censi et al., 2019). However, it did not include the highly non-linear behavior of contacts which are ubiquitous in manipulation and difficult to learn. All other robotic challenges (e.g. *REAL* (Cartoni et al., 2020), *MineRL* (Kanervisto et al., 2022) and *Learn to Move* (Song et al., 2021)) were restricted to simulations. Unfortunately, the policies learned in simulation often do not transfer to the real world.

The *Real Robot Challenge II* (Bauer et al., 2022), hosted last year at NeurIPS, was the first challenge in the NeurIPS competition track that is geared towards learning methods for control on real robots in a fully remote setup. However, in the previous instantiations of the

2. https://nips.cc/Conferences/2019/CompetitionTrack
3. https://neurips.cc/Conferences/2020/CompetitionTrack

Real Robot Challenge, there were no restrictions on the algorithms used for controlling the robot.

In this year's competition, we exclusively focused on the offline RL paradigm. The goal of offline RL is to learn effective policies from large and diverse datasets covering a sufficient amount of expert transitions without additional online interaction (Levine et al., 2020). Although several algorithmic advances have been proposed in offline RL in recent years, a standardized benchmark of real-word robotic data has not been established yet. As featured in Mandlekar et al. (2021), there exist small real-world datasets with human demonstrations for a robot arm with a gripper (using operational space control). However, a dataset sufficiently large for offline RL with low-level control to solve more challenging manipulation tasks had been missing. In our challenge, we have provided one such benchmarking dataset that can easily be evaluated remotely on a real-robot platform. This benchmark dataset has also been featured in our concurrent work (Gürtler et al., 2023).

3. Challenge

The goal of the Real Robot Challenge 2022 was to solve manipulation tasks with TriFinger robots by learning solely from pre-recorded datasets, without access to additional online interactions.

3.1. Tasks and Stages

We considered two dexterous manipulation tasks involving a tracked cube (see Fig. 1, right):

Push The goal is to push the cube to a target position which is sampled from a uniform distribution on the ground of the arena. The orientation of the cube does not influence the reward in this task.

Lift For the Lift task a target position in the air and a target orientation have to be matched. The target position is sampled up to a height of 10 cm such that the desired cube pose does not intersect with the ground. The desired orientation is sampled uniformly.

The Lift task is significantly more challenging than the Push task as it requires flipping the cube to an approximately correct orientation, acquiring a stable grasp, lifting it to the goal position and turning it in-hand to match the target orientation. If the cube slips from the fingers all progress is usually lost. This renders the Lift task – together with the noise on the pose estimation of the cube – quite unforgiving.

We calculate the reward by applying a logistic kernel to the difference between desired and achieved position (for the Push task) or to the differences between the desired and achieved corner points of the cube (for the Lift task) similar to Allshire et al. (2022). This choice results in a smooth falloff of the reward when deviating from the goal and does not require manually balancing the influence of position and orientation. Further details can be found in Appendix D.

We divided the challenge into two overlapping stages:

Pre-stage (July 1 to September 1, 2022): The pre-stage served as an open qualification round in which everybody could participate. The objective was to learn proficient policies for the Push and Lift tasks from provided simulated datasets containing expert trajectories. The

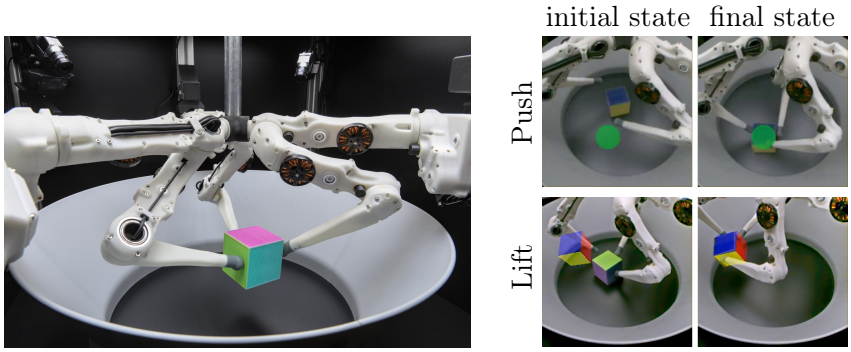


Figure 1: Left: TriFingerPro robot holding a cube. Right: Examples of initial and final states of successful episodes of the Push and Lift tasks.

submissions were then evaluated in a simulated version of the TriFinger platform. Teams that reached a promising level of performance on both tasks were admitted to the next stage.

Real-robot stage (August 1 to October 7, 2022): With the start of the real-robot stage we released four datasets recorded on real TriFinger robots. All qualifying teams were provided with remote access to the robot cluster (see section 3.4). For each task, two policies had to be learned separately from two datasets with different compositions (see section 4).

3.2. Hardware

For the real-robot phase we used the same *TriFingerPro* robot cluster that was already used in the previous iterations of the competition (Bauer et al., 2022).

The *TriFingerPro* robots consist of three fingers that can pick up and manipulate objects within a circular arena (see left side of Fig. 1). They are an enhancement of the open-source *TriFingerEdu* (Wüthrich et al., 2020) to make them even more robust and thus reduce the maintenance effort. The cluster consists of seven such robots that can be accessed remotely (see section 3.4). The joints can be torque-controlled through electric motors. Push sensors on the finger tips can be used to detect contact. Each robot platform is further equipped with three RGB cameras that are used to estimate the pose of the manipulated object.

The robots are designed to be able to operate 24/7 without human supervision. This is made possible by the robust hardware design, several safety measures in the software (e.g. limits on torques or position range of the joints), automated self-tests after each run and the ability to reset the object position autonomously.

3.3. Rules

The participants were allowed to use any method to learn policies from the provided datasets as long as they complied with the following rules[4]:

4. The complete list of rules for the competition (including technicalities) is given in Appendix B.

- Policies had to be learned exclusively from one dataset at a time, i.e., combining datasets or training on additional data (simulated or real) was not allowed.
- The training code had to be released under an OSI-approved license and a report describing the method had to be published in a publicly accessible way (e.g. on arXiv).

The final ranking of the teams was determined by their submissions in the real-robot stage, as the pre-stage served only as a qualifying round. The main criterion was the score obtained by the submitted policies (see section 3.5). For teams which were closely matched in terms of scores, we also considered the quality of the published report as a secondary criterion for the final ranking.

Due to generous support by DeepMind, we were able to award the following sums as prize money to the highest ranking teams: (1) 2500 USD (2) 1500 USD (3) 1000 USD (split into two times 500 USD as the third rank was shared by two teams with similar results).

3.4. Submission System

In the real-robot stage, participants got access to the submission system of our robot cluster. This system allows users to remotely submit jobs with their policies to the cluster, which are then automatically executed on a randomly selected robot. The resulting data can then be downloaded by the user, once the job is finished.

Compared to the previous challenges (Bauer et al., 2022), two major changes were made:

1. Participants only implemented a policy using a given interface in Python. The actual control loop was implemented on the server side and could not be modified by the participants (apart from a few configuration options like specifying the task).
2. Since the task was to learn policies from the given datasets, participants were not allowed to collect additional data for the training. Therefore the recorded sensor data was not provided to participants. They only received the resulting score and related statistics as well as a video of their runs.

One job has a runtime of around four minutes, so multiple episodes can be executed within one job, depending on the task. For the push task nine episodes are executed, for the lift task (which has a longer episode length) only six. Between each episode a short "object reset trajectory" is executed to bring the cube back towards the center of the arena.

3.5. Evaluating Submissions

Pre-stage

Solutions for the pre-stage were submitted via a form and evaluated in the simulated environment. The same evaluation protocol as in the real-robot stage (see below) was used except for sampling random goals instead of using a fixed sequence.

Real-robot stage

For evaluating the policies of the participants we used the following procedure:

1. Get the current code of all teams from the submission system.
2. Select N_R robots to be used for the evaluation. For each robot generate a list of N_G random goals for each task/dataset combination.
3. Run the code of all teams on these goals.

For each episode a score is computed as the cumulative reward over all time steps. For each task/dataset combination the mean score of the episodes of all corresponding runs is computed.

The total score for the ranking is then the mean over all task/dataset combinations. Note that the episode length of the lift task is longer than that of the push task. Since the score is the unnormalised cumulative reward over all time steps, this means that the scores for the lift task tend to be higher than those of the push tasks. This results in the lift task getting a higher weight in the total score computation, which is, however, intended as it is the more challenging task.

4. Data Collection

We collected datasets on 6 real TriFinger robots and in a simulated PyBullet environment (Joshi et al., 2020) which was closely modeled after the real system. We used policies from Gürtler et al. (2023) which were trained with Proximal Policy Optimization (Schulman et al., 2017). The training pipeline of Gürtler et al. (2023) builds upon the work of Allshire et al. (2022), which uses a fast, GPU-based rigid body physics simulator (Makoviychuk et al., 2021) to parallelize rollouts. In addition to the converged expert policies, we also consider an early training checkpoint with additive noise on the actions to which we refer as *weak* policy.

We provided datasets with two different compositions for each of the two tasks: (i) The *expert* dataset consists solely of trajectories collected with the converged policies and tests the ability to imitate a proficient behavior policy. (ii) In contrast to this, half of the *mixed* dataset consists of trajectories obtained with the weak policy while the other half contains expert trajectories. Learning a good policy from this dataset requires a training algorithm that either performs credit assignment or distills high-quality trajectories to imitate. Table 2 summarizes the six datasets used in the competition.

5. Methods

We present the participants' solutions in the context of state-of-the-art offline RL methods to which we compare quantitatively in section 6.

5.1. State-of-the-Art Algorithms

In the following, we briefly summarize a selection of offline RL algorithms: Behavioral Cloning (BC) (Bain and Sammut, 1995; Pomerleau, 1991; Ross et al., 2011; Torabi et al., 2018) is a purely supervised method in which the mean squared error between the actions of the behavioral policy $a \sim \pi_\beta(\cdot \mid s)$ and the learned policy $a \sim \pi(\cdot \mid s)$ is minimized. Critic Regularized Regression (CRR) (Wang et al., 2020) is a BC variant in which the actions

are *weighted* according to advantage-based estimates using Q-values. CRR optimizes the following objective:

$$\arg\max_{\pi} \mathbb{E}_{(s,a)\sim\mathcal{B}} \left[f(Q_\theta, \pi, s, a) \log \pi(s, a) \right],$$

where f is a non-negative, scalar function whose value is monotonically increasing in Q_θ. Advantage Weighted Actor Critic (AWAC) (Nair et al., 2020) is an actor-critic method in which the policy improvement step is formulated as a constrained optimization problem that forces the policy to stay close to the behavioral policy. Conservative Q-Learning (CQL) (Kumar et al., 2020) is an actor-critic method that combines the Bellman update in the critic loss with a conservative loss that aims to push down Q-values of out-of-distribution (OOD) actions. Implicit Q-Learning (IQL) (Kostrikov et al., 2021) is an offline RL algorithm that avoids out-of-distribution action queries during training. It mitigates overshooting of the value function by estimating a Q-function expectile, and then performs policy extraction with weighted Behavioral Cloning. For more details on Offline RL, we refer the reader to the surveys Levine et al. (2020); Prudencio et al. (2022).

5.2. Team "excludedrice" (1st place)

Qiang Wang and Robert McCarthy – University College Dublin

The training of the robot controller used by team *excludedrice* is based on BC. Their full solution can be found at (Wang et al., 2023b). Simply put, in their work, they found that BC performed better when cloning expert demonstrations than when training with complex offline reinforcement learning applied to data containing demonstrations with mixed skill levels. Nevertheless, BC tends to perform poorly on mixed datasets that contain mixed skill levels, which can introduce ambiguity and make it more difficult for BC's supervised learning process to accurately perform the required regression. In general, BC is best suited to situations where the actions being modeled are conditioned on states from a unimodal distribution, or when the target action mode makes up the majority of the data in the dataset. After investigating the composition of the mixed quality datasets, they discovered that half of the data was collected by experts, and this subset of data was potentially adequate for training a good policy. Thus, their objective was to filter out this expert data for training a controller using BC. However, simple manual methods were unable to differentiate the expert data as the performance of both expert and non-expert data was similar. As a result, they proposed a novel semi-supervised learning data filtering approach in their strategy. Initially, they extracted a small portion of the data with the highest scores from the entire dataset that was presumed to be mostly collected by an expert agent. It is worth noting that the size of this initial extracted data subset was insufficient to train a well-performing policy model. They fed this portion of data into a neural network (NN) to learn patterns from expert data, and then used it as a binary classifier to separate out more expert data for training the next iteration of the NN. They repeated this semi-supervised learning process iteratively until the number of separated expert data no longer increased. They improved this algorithm after the competition, and their methodology can be found in Wang et al. (2023a).

Furthermore, they augmented the training data of the robotic arena using spatial rotation transformations, taking advantage of the rotational symmetry of the physical TriFinger robot. However, it is important to note that data augmented through mathematical theory may

not be entirely accurate in real-world scenarios. For instance, factors like friction and the dimensions of different fingers of the robot may not be identical due to physical errors during manufacturing. To address this issue, they proposed a policy training paradigm to make the model trained on theoretical data to better fit the data distribution of the real robotic environment.

5.3. Team "decimalcurlew" (2nd place)

Hangyeol Kim and Jongchan Baek Pohang – Pohang University of Science and Technology (POSTECH)
Wookyong Kwon – Electronics and Telecommunications Research Institute (ETRI)

Team *decimalcurlew* used offline RL and a regularization technique to obtain robust policies that perform well in a real robot system, even with measurement noise. They trained a feed-forward neural network policy for each task with nonlinear rectified linear units and 400 and 300 hidden units. The team preprocessed the dataset for training by performing state and action normalization, and scaling the actions to fit within the range of $[-1, 1]$. They then trained the policy networks on the normalized dataset using the offline RL algorithm TD3+BC Fujimoto and Gu (2021).

Given the uncertainty of observation noise in a real-robot system, the team aimed to obtain policies that could work robustly against such noise. To accomplish this, they adapted the policy training objective of TD3+BC and added a regularization term that encourages the policy's actions to be spatially smooth Mysore et al. (2021), leading to similar actions for comparable states in the robot system. In the final stage of the challenge, their policies exhibited competitive performance across all tasks.

5.4. Team "superiordinosaur" (shared 3rd place)

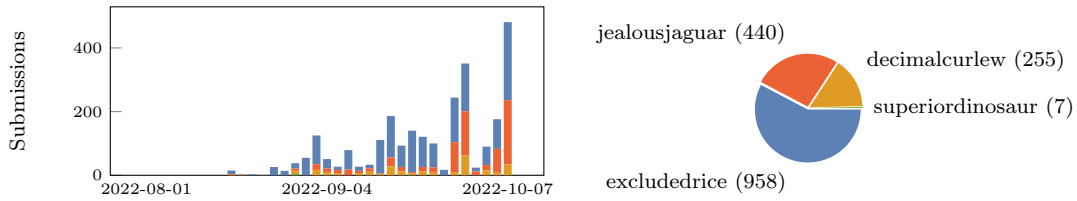
Shanliang Qian – Independent

Team *superiordinosaur* observed an important feature of the real dataset, the vision tracking system is far from perfect and sometimes suffers from: i) high delay and ii) noisy cube pose estimation. They proposed a simple approach that combines supervised learning, early stopping and the introduction of a validity check with a smoothing process that maintains a moving average of the cube pose. In particular, for a new cube pose, their algorithm checks whether the delay is less than or the confidence is greater than certain thresholds, in order to decide whether to update the cube pose with a moving average or to keep the last cube pose instead. The team also report unsatisfactory results with the TD3+BC algorithm and LSTM architectures.

5.5. Team "jealousjaguar" (shared 3rd place)

Yasunori Toshimitsu, Mike Yan Michelis, Amirhossein Kazemipour, Arman Raayatsanati, Hehui Zheng, and Barnabasa Gavin Cangan – ETH Zurich

Team *jealousjaguar* used the offline RL algorithm "Implicit Q-Learning" (IQL) Kostrikov et al. (2021), due to its ability to avoid out-of-distribution action queries during training



(*a*) Over time (one bar = two days) (*b*) Per team

Figure 2: Number of job submissions to our robots over time and per team.

and to mitigate a value function overshooting by estimating a Q-function expectile via an asymmetric ℓ_2 loss. They selected the IQL implementation provided by the open source library d3rlpy (Seno and Imai, 2021).

The team has furthermore created an automation pipeline that allows submissions to be queued, and sent to the real robot cluster automatically (the RRC system only accepted submissions if there was no ongoing submission by the same team), which allowed the performance of the policy to be gauged periodically during training and uploaded to Weights & Biases (Biewald, 2020). This made the comparison of different algorithms and parameters easier, allowing the team to develop their own ideas and compare in real time with others. They also introduced various methods to improve the performance and consistency of a policy, such as data augmentation, or the inclusion of previous observations and actions in the state.

6. Results

We present the results of the real-robot stage in this section and compare them to what state-of-the-art offline RL algorithms can achieve on the challenge datasets. The results of the pre-stage are summarized in Appendix C.

6.1. Usage Statistics

Throughout the real-robot stage, the participating teams submitted a total of 1660 jobs to the robots. Moreover, the number of submitted jobs differed highly between teams. Figure 2 shows the distribution of jobs over time and teams.

6.2. Results

Table 1 shows the average returns the teams achieved on all task/dataset combinations in the real-robot phase. The last column contains the overall score which is obtained by averaging these returns. We furthermore include the relevant benchmarking results from Gürtler et al. (2023) as a point of reference[5]. The scores of the teams are compared to those of the benchmarked offline RL algorithms in Fig. 3. We additionally provide success rates in Table 4 in appendix C .

Team excludedrice achieved the highest score by a significant margin by combining self-supervised dataset filtering with Behavioral Cloning. This approach even outperformed the

5. Note that the Mixed datasets correspond to the Weak&Expert datasets in Gürtler et al. (2023).

Table 1: **Returns and overall score in the real-robot stage:** For each combination of task and behavior policy the mean return and the standard error of the mean return are given (failed runs correspond to a return of 0). The overall score is the return averaged over all tasks.

		Push/Expert	Push/Mixed	Lift/Expert	Lift/Mixed	Score
	behavior policies	660 ± 2	429 ± 4	1064 ± 7	851 ± 8	751 ± 3
	Teams					
1.	excludedrice	624 ± 6	**635 ± 5**	**956 ± 21**	**923 ± 21**	**784 ± 8**
2.	decimalcurlew	**639 ± 4**	613 ± 5	841 ± 20	717 ± 18	703 ± 7
3.	superiordinosaur	618 ± 6	575 ± 8	856 ± 22	571 ± 17	655 ± 7
	jealousjaguar	**639 ± 5**	561 ± 7	855 ± 19	506 ± 17	640 ± 7
	Algorithms	(results from Gürtler et al. (2023))				
	BC	562 ± 14	388 ± 21	676 ± 35	437 ± 26	516 ± 13
	CRR	638 ± 8	621 ± 11	890 ± 34	707 ± 28	714 ± 12
	AWAC	623 ± 9	567 ± 14	747 ± 36	481 ± 28	605 ± 12
	CQL	514 ± 15	346 ± 17	288 ± 16	269 ± 14	354 ± 8
	IQL	592 ± 10	555 ± 14	900 ± 32	574 ± 31	655 ± 12

behavior policy on the challenging mixed data from the Lift task, unlike all other competitors. As a result, excludedrice is the only team that exceeds the score of the behavior policies.

On the second rank, decimalcurlew also achieves good results on the Lift-Mixed dataset with a regularized version of TD3+BC. In contrast to this, the remaining teams fall behind on this decisive dataset. Surprisingly, the BC-based approach of team superiordinosaur slightly outperforms team jealousjaguar's solution built around the offline RL algorithm IQL. This may be a result of team superiordinosaur's effort to take the confidence of the tracking system into account when updating the pose estimate which seems to result in better performance on the Lift-Mixed dataset.

Two of the offline RL algorithms benchmarked in Gürtler et al. (2023) achieved scores comparable to some of the top submissions. CRR reaches a score similar to that of decimalcurlew while IQL matches the score of superiordinosaur. Note, however, that the hyperparameters of the benchmarked algorithms used for the Lift task were optimized on the simulated version of Lift-Mixed. This requires a significant amount of computational resources but increases the scores on the real Lift-Mixed dataset.

6.3. Challenge Takeaways

Imitation learning vs offline RL: In principle, offline RL should be able to outperform imitation learning on datasets containing suboptimal trajectories (which is the case for all challenge datasets), as it takes the reward signal into account. We were therefore surprised that two out of four top teams built their methods around Behavioral Cloning. We see several factors that could contribute to the popularity of BC: (i) Offline RL algorithms are

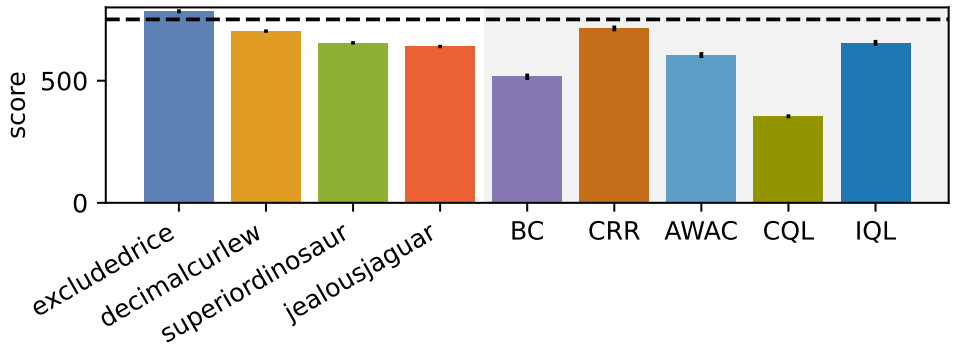


Figure 3: **Scores in the real-robot stage:** Overall scores of the winning teams and state-of-the-art offline RL algorithms for comparison. The averaged score of the behavior policies is indicated by a dashed line.

sometimes difficult to implement and tune, unlike BC, (ii) the robotics community might not have fully adopted the offline RL paradigm yet, and (iii) in practice other parts of the method might have a bigger impact on performance.

Algorithmic vs. problem-specific adaptations: Instead of optimizing the choice of learning algorithm or the algorithm itself, most teams concentrated on orthogonal contributions like filtering, data augmentation and regularization which were partly tailored to the robotics problem. They reported significant improvements in performance following this strategy.

Simulation vs. real world: The gap between expert policy and learned policies was, on average, bigger on real-world data. This may be caused by more complex real-world dynamics as also discussed in Gürtler et al. (2023).

7. Conclusion

In summary, the competition provided an opportunity to apply offline RL where it matters most: in the real world. The results show somewhat surprisingly that simpler methods, such as Behavioral Cloning combined with suitable filtering and data augmentation, can be more effective in real-world applications than more elaborate offline RL algorithms. This in turn means that more research on offline RL is required and that new algorithms should be also evaluated on real hardware (Gürtler et al., 2023).

Acknowledgments

We thank Thomas Steinbrenner for the maintenance of the robot platforms. We are furthermore grateful for the financial support of DeepMind that made the prize money possible.

GM and BS are members of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645. This work was supported by the Volkswagen Stiftung

(No 98 571). We acknowledge the support from the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039B).

References

Arthur Allshire, Mayank MittaI, Varun Lodaya, Viktor Makoviychuk, Denys Makoviichuk, Felix Widmaier, Manuel Wüthrich, Stefan Bauer, Ankur Handa, and Animesh Garg. Transferring dexterous manipulation from GPU simulation to a remote real-world trifinger. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11802–11809, 2022. URL https://ieeexplore.ieee.org/document/9981458.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In Koichi Furukawa, Donald Michie, and Stephen H. Muggleton, editors, *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, UK, July 1995]*, pages 103–129. Oxford University Press, 1995.

Stefan Bauer, Manuel Wüthrich, Felix Widmaier, Annika Buchholz, Sebastian Stark, Anirudh Goyal, Thomas Steinbrenner, Joel Akpo, Shruti Joshi, Vincent Berenz, Vaibhav Agrawal, Niklas Funk, Julen Urain De Jesus, Jan Peters, Joe Watson, Claire Chen, Krishnan Srinivasan, Junwu Zhang, Jeffrey Zhang, Matthew Walter, Rishabh Madan, Takuma Yoneda, Denis Yarats, Arthur Allshire, Ethan Gordon, Tapomayukh Bhattacharjee, Siddhartha Srinivasa, Animesh Garg, Takahiro Maeda, Harshit Sikchi, Jilong Wang, Qingfeng Yao, Shuyu Yang, Robert McCarthy, Francisco Sanchez, Qiang Wang, David Bulens, Kevin McGuinness, Noel O'Connor, Redmond Stephen, and Bernhard Schölkopf. Real robot challenge: A robotics competition in the cloud. In *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, volume 176 of *Proceedings of Machine Learning Research*, pages 190–204. PMLR, 06–14 Dec 2022. URL https://proceedings.mlr.press/v176/bauer22a.html.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Emilio Cartoni, Francesco Mannella, Vieri Giuliano Santucci, Jochen Triesch, Elmar Rueckert, and Gianluca Baldassarre. Real-2019: Robot open-ended autonomous learning competition. In *NeurIPS 2019 Competition and Demonstration Track*, pages 142–152. PMLR, 2020. URL http://proceedings.mlr.press/v123/cartoni20a.

Andrea Censi, Liam Paull, Jacopo Tani, and Matthew R Walter. The AI driving olympics: An accessible robot learning benchmark. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019. URL https://doi.org/10.3929/ethz-b-000464062.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning, 2020. URL https://arxiv.org/abs/2003.11881.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. URL https://arxiv.org/abs/2004.07219.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, pages 20132–20145, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/a8166da05c5a094f7dc03724b41886e5-Abstract.html.

Nico Gürtler, Sebastian Blaes, Pavel Kolev, Felix Widmaier, Manuel Wuthrich, Stefan Bauer, Bernhard Schölkopf, and Georg Martius. Benchmarking offline reinforcement learning on real-robot hardware. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3k5CUGDLNdd.

Shruti Joshi, Felix Widmaier, Vaibhav Agrawal, and Manuel Wüthrich. https://github.com/open-dynamic-robot-initiative/trifinger_simulation, 2020.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor Sampedro, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning 2018*, 2018. URL https://arxiv.org/pdf/1806.10293.

Anssi Kanervisto, Stephanie Milani, Karolis Ramanauskas, Nicholay Topin, Zichuan Lin, Junyou Li, Jianing Shi, Deheng Ye, Qiang Fu, Wei Yang, et al. Minerl diamond 2021 competition: Overview, results, and lessons learned. *NeurIPS 2021 Competitions and Demonstrations Track*, pages 13–28, 2022. URL https://proceedings.mlr.press/v176/kanervisto22a.html.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *CoRR*, abs/2110.06169, 2021. URL https://arxiv.org/abs/2110.06169.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. *Reinforcement learning: State-of-the-art*, pages 45–73, 2012. URL https://doi.org/10.1007/978-3-642-27645-3_2.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020. URL https://arxiv.org/abs/2005.01643.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. 2021. URL https://arxiv.org/abs/2108.10470.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 1678–1690. PMLR, 2021. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing action policies for smooth control with reinforcement learning. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 1810–1816. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9561138. URL https://doi.org/10.1109/ICRA48506.2021.9561138.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *CoRR*, abs/2006.09359, 2020. URL https://arxiv.org/abs/2006.09359.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand. https://arxiv.org/abs/1910.07113, October 2019.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991. URL https://doi.org/10.1162/neco.1991.3.1.88.

Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *CoRR*, abs/2203.01387, 2022. doi: 10.48550/arXiv.2203.01387. URL https://doi.org/10.48550/arXiv.2203.01387.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org, 2011. URL http://proceedings.mlr.press/v15/ross11a/ross11a.pdf.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 91–100. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/rudin22a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *CoRR*, abs/2111.03788, 2021. URL https://arxiv.org/abs/2111.03788.

Seungmoon Song, Łukasz Kidziński, Xue Bin Peng, Carmichael Ong, Jennifer Hicks, Sergey Levine, Christopher G Atkeson, and Scott L Delp. Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation. *Journal of neuroengineering and rehabilitation*, 18:1–17, 2021. URL https://doi.org/10.1186/s12984-021-00919-y.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4950–4957. ijcai.org, 2018. doi: 10.24963/ijcai.2018/687. URL https://doi.org/10.24963/ijcai.2018/687.

Qiang Wang, Robert McCarthy, David Cordova Bulens, Kevin McGuinness, Noel E O'Connor, Francisco Roldan Sanchez, and Stephen J Redmond. Behaviour discriminator: A simple data filtering method to improve offline policy learning. https://arxiv.org/abs/2301.11734, 2023a.

Qiang Wang, Robert McCarthy, David Cordova Bulens, and Stephen J Redmond. Winning solution of real robot challenge iii. https://arxiv.org/abs/2301.13019, 2023b.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh Merel, Jost Tobias Springenberg, Scott E. Reed, Bobak Shahriari, Noah Y. Siegel, Çaglar Gülçehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/588cb956d6bbe67078f29f8de420a13d-Paper.pdf.

Manuel Wüthrich, Felix Widmaier, Felix Grimminger, Joel Akpo, Shruti Joshi, Vaibhav Agrawal, Bilal Hammoud, Majid Khadiv, Miroslav Bogdanovic, Vincent Berenz, et al. Trifinger: An open-source robot for learning dexterity. *Conference on Robot Learning (CORL)*, 2020. URL https://proceedings.mlr.press/v155/wuthrich21a.html.

Appendix A. Datasets

We provide an overview of the datasets provided to the participants in table 2.

Appendix B. Complete set of rules

The complete set of rules of the Real Robot Challenge 2022 was as follows:

- Any algorithmic approach may be applied that learns the behavior only from the provided data and does not make use of any hard-coded/engineered behavior. As an example, two prominent algorithmic approaches meeting this criteria are: offline reinforcement learning and imitation learning.

Table 2: Overview of the offline RL datasets provided to the participants.

task	dataset	overall duration [h]	#episodes	#transitions [10^6]	episode length [s]
Push-	Sim-Expert	16	3840	2.8	15
	Real-Expert	16	3840	2.8	15
	Real-Mixed	16	3840	2.8	15
Lift-	Sim-Expert	20	2400	3.6	30
	Real-Expert	20	2400	3.6	30
	Real-Mixed	20	2400	3.6	30

- It is not permitted to use data collected during evaluation rollouts or obtained from other sources.
- It is not permitted to use data provided for one task to train a policy for an other task (e.g. use simulation data for the real robot or the "expert" dataset for the "mixed" task).
- It is not permitted to filter the datasets based on the position of a sample in the dataset. However, you may filter based on the properties of a transition or an episode if you want to.
- Participants may participate alone or in teams.
- Individuals are not allowed to participate in multiple teams.
- Each team needs to nominate a contact person and provide an email address through which they can be reached.
- Cash prizes will be paid out to an account specified by the contact person of each team. It is the responsibility of the team's contact person to distribute the prize money according to their team-internal agreements.
- To be eligible to win prizes, participants agree to release their code under an OSI-approved license and to publish a report describing their method in a publicly accessible way (e.g. on arXiv).
- Participants may not alter parameters of the simulation (e.g. the robot model) for the evaluation of the pre-stage.
- The organizers reserve the right to change the rules if doing so is absolutely necessary to resolve unforeseen problems.
- The organizers reserve the right to disqualify participants who are violating the rules or engage in scientific misconduct.

Appendix C. Additional results

Table 3: **Returns in the pre-stage** which were obtained by evaluating in the simulated environment after training on datasets recorded in simulation. The teams are listed as they ranked in the real-robot stage.

	Push/Expert	Lift/Expert
behavior policies	674	1334
Teams		
excludedrice	676	1273
decimalcurlew	675	1132
superiordinosaur	653	1325
jealousjaguar	658	1137

Table 4: **Success rates in the real-robot stage:** For each combination of task and behavior policy the mean success rate and the standard error of the mean success rate are given (failed runs correspond to a return of 0). The last column is the success rate averaged over all datasets.

		Push/Expert	Push/Mixed	Lift/Expert	Lift/Mixed
	behavior policies	0.92 ± 0.01	0.51 ± 0.01	0.66 ± 0.01	0.40 ± 0.01
	Teams				
1.	excludedrice	0.82 ± 0.02	0.83 ± 0.01	0.48 ± 0.02	$\mathbf{0.46 \pm 0.02}$
2.	decimalcurlew	0.80 ± 0.02	0.71 ± 0.02	0.28 ± 0.02	0.13 ± 0.02
3.	superiordinosaur	0.80 ± 0.02	0.69 ± 0.02	0.40 ± 0.02	0.11 ± 0.02
	jealousjaguar	0.79 ± 0.02	0.59 ± 0.02	0.25 ± 0.02	0.03 ± 0.01
	Algorithms	(results by Gürtler et al. (2023))			
	BC	0.74 ± 0.02	0.48 ± 0.03	0.28 ± 0.02	0.09 ± 0.02
	CRR	$\mathbf{0.87 \pm 0.03}$	$\mathbf{0.84 \pm 0.04}$	$\mathbf{0.54 \pm 0.04}$	0.29 ± 0.04
	AWAC	0.80 ± 0.01	0.69 ± 0.03	0.31 ± 0.02	0.12 ± 0.03
	CQL	0.54 ± 0.06	0.14 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	IQL	0.75 ± 0.03	0.68 ± 0.03	0.48 ± 0.03	0.15 ± 0.01

Appendix D. Reward function and success criterion

The reward is obtained by applying the logistic kernel

$$k(x) = (b+2)\left(\exp(a\|x\|) + b + \exp(-a\|x\|)\right)^{-1} \tag{1}$$

to the difference between desired and achieved position (for the Push task) or the desired and achieved corner points of the cube (for the Lift task). The parameters a and b control the length scale over which the reward decays and how sensitive it is for small distances x, respectively.

We consider an episode successful if at its end the desired position is matched up to a tolerance of 2 cm and the deviation from the desired orientation does not exceed 22 deg, similar to Allshire et al. (2022) and Gürtler et al. (2023).

Appendix E. Code repositories of the winning teams

Teams	URL to repository
excludedrice	https://github.com/wq13552463699/Real-Robot-Challenge-2022.git
decimalcurlew	https://github.com/paekgga/RRC2022Training
superiordinosaur	https://github.com/QianSL/rrc_solution
jealousjaguar	https://github.com/QianSL/rrc_solution

Table 5: URLs of the code repositories of the winning teams.