Dispelling the Mirage of Progress in Offline MARL through Standardised Baselines and Evaluation

Claude Formanek[1,2*] **Callum Rhys Tilbury**[1] **Louise Beyers**[1]

Jonathan Shock[2,3,4] **Arnu Pretorius**[1]

[1]InstaDeep [2]University of Cape Town [3]INRS, Montreal [4]NITheCS, Stellenbosch

Abstract

Offline multi-agent reinforcement learning (MARL) is an emerging field with great promise for real-world applications. Unfortunately, the current state of research in offline MARL is plagued by inconsistencies in baselines and evaluation protocols, which ultimately makes it difficult to accurately assess progress, trust newly proposed innovations, and allow researchers to easily build upon prior work. In this paper, we firstly identify significant shortcomings in existing methodologies for measuring the performance of novel algorithms through a representative study of published offline MARL work. Secondly, by directly comparing to this prior work, we demonstrate that simple, well-implemented baselines can achieve state-of-the-art (SOTA) results across a wide range of tasks. Specifically, we show that on 35 out of 47 datasets used in prior work (almost 75% of cases), we match or surpass the performance of the current purported SOTA. Strikingly, our baselines often substantially outperform these more sophisticated algorithms. Finally, we correct for the shortcomings highlighted from this prior work by introducing a straightforward standardised methodology for evaluation and by providing our baseline implementations with statistically robust results across several scenarios, useful for comparisons in future work. Our proposal includes simple and sensible steps that are easy to adopt, which in combination with solid baselines and comparative results, could substantially improve the overall rigour of empirical science in offline MARL moving forward.

1 Introduction

Offline reinforcement learning (RL) attempts to derive optimal sequential control policies from static data alone, without access to online interactions (e.g. a simulator). Though the single-agent variant has received fairly widespread research attention (Prudencio et al., 2023), progress in the multi-agent context has been slower, due to a variety reasons. For one, multi-agent problems are fundamentally more difficult, and bring a host of new challenges—including difficulties in coordination (Barde et al., 2024), large joint-action spaces (Yang et al., 2021), heterogeneous agents (Zhong et al., 2024) and non-stationarity (Papoudakis et al., 2019), which are absent in single-agent situations.

Nonetheless, some progress has been made in offline multi-agent reinforcement learning (MARL) in recent years. In particular, in better understanding the aforementioned difficulties of offline learning in the multi-agent setting and proposing certain remedies for them (Jiang and Lu, 2021; Yang et al., 2021; Pan et al., 2022; Wang et al., 2023; Shao et al., 2023; Tian et al., 2023; Meng et al., 2023).

*Corresponding author: `c.formanek@instadeep.com`

Preprint. Accepted at NeurIPS 2024 Datasets & Benchmarks Track.

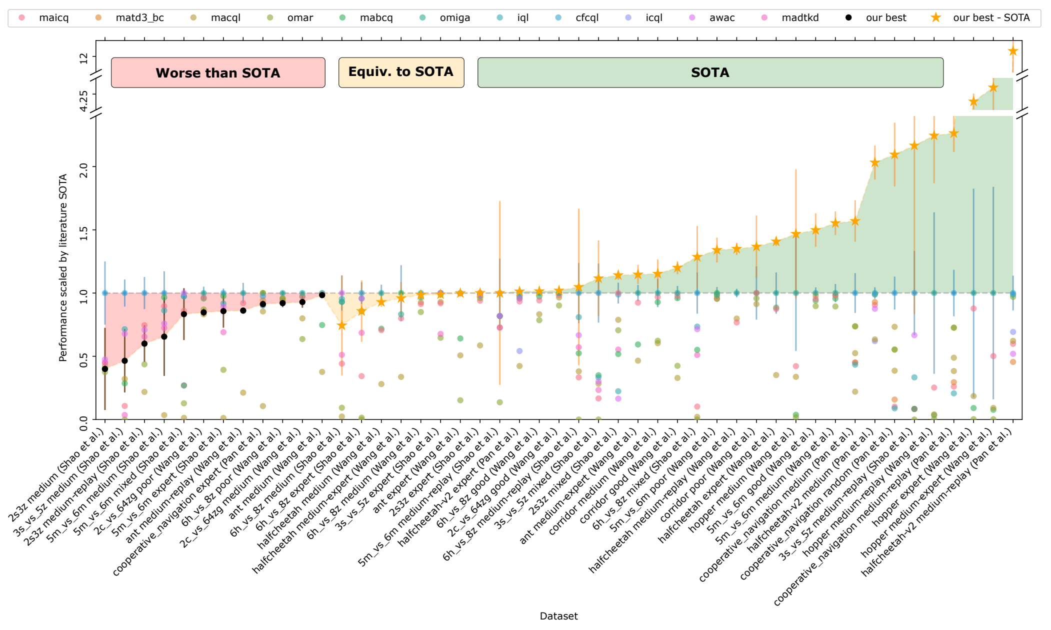


Figure 1: We compare our baseline implementations to the reported performance of various algorithms from the literature across a wide range of datasets. We normalise results from each dataset (i.e. scenario-quality-source combination) by the SOTA performance from the literature for that dataset. Standard deviation bars are given and when our baseline is significantly better or equal to the best method, using a two-side t-test, we indicate so using a gold star. **We find that on 35 out of the 47 datasets tested (almost 75% of cases), we match or surpass the performance of the current SOTA.**

However, we argue that this progress might be a mirage and that offline MARL research is ultimately being held back by a lack of clarity and consistency in baseline implementations and evaluation protocols. To support this claim, we demonstrate in Figure 1 a surprising but telling result—we show that good implementations of straightforward baseline algorithms can achieve state-of-the-art performance across a wide-range of tasks, beating several published works claiming such a title. We view our analysis as robust, using datasets and environments with experimental settings that exactly match prior work (see Section 3).

This paper proceeds as follows. We first assess the state of the field, diagnosing the key points of friction for progress in offline MARL. Thereafter, we describe in more detail how well-implemented baseline methods perform surprisingly well compared to leading methods from the literature, suggesting that algorithmic progress has not advanced at the rate perhaps previously perceived by the community. In response, we introduce a standardised baseline and evaluation methodology, in an effort to support the field towards being more scientifically rigorous. We hope that researchers will build upon this work, advocating for a cleaner, more reproducible and robust empirical science for offline MARL.

2 Methodological Problems in Offline MARL

In this section, we briefly assess the state of offline MARL research by focusing on the baselines and evaluation protocols commonly employed. We consider the following five papers, all published at top-tier venues, for our case study: MAICQ (Yang et al., 2021), OMAR (Pan et al., 2022), MADT (Meng et al., 2023), CFCQL (Shao et al., 2023), and OMIGA (Wang et al., 2023). Given the nascency of the field, we consider these papers to serve as a good representative sample of the current trends in offline MARL research. By looking at this cross-section, we can assess the current methodologies for measuring progress. We present our findings below.

Table 1: Demonstration of how papers in our case study are essentially using the same name for Multi-Agent CQL, for markedly different algorithms, often providing only sparse information about their implementations.

Paper	Name for `MACQL`	Implementation Details Provided in the Paper	Is the Full Code Available?
Yang et al. (2021)	MA-CQL	Loss function, value-decomposition structure	No, only single-agent CQL
Pan et al. (2022)	MA-CQL	MADDPG (Lowe et al., 2017) (Discrete: + Gumbel Softmax (Jang et al., 2016))	Only for continuous settings
Meng et al. (2023)	CQL-MA	CQL + "mixing network"	No
Shao et al. (2023)	MACQL	"Naive extension of CQL to multi-agent settings" + Loss function	Yes
Wang et al. (2023)	CQL-MA	Value decomposition structure + policy constraint	No

Ambiguity in the Naming of Baseline Algorithms In single-agent RL, the naming of a given algorithm is fairly unambiguous: it is widely understood what core algorithmic steps constitute, e.g., DDPG (Lillicrap et al., 2016). Yet, in MARL, algorithms become more complex, since one must specify how multiple agents should learn and interact. Unlike in the single agent case, there is ambiguity when referring to *multi-agent* DDPG—for this might be referring to MADDPG (Lowe et al., 2017), or independent DDPG agents, or perhaps some other way of interleaving training and/or execution with DDPG forming the base of the algorithm's design. The corresponding impact on the performance of such choices can be significant (Lyu et al., 2021), and thus it is important to be as explicit as possible.

Clarity in this naming has been lacking in offline MARL literature. For instance, we consider Conservative Q-Learning (CQL) (Kumar et al., 2020) as a prime example of this problem. As an influential single-agent offline RL algorithm, CQL is critical to consider as a baseline when proposing new work in the field. Whereas the core CQL algorithmic steps are well-established, though, there does not exist a common understanding in the literature of what *multi-agent* CQL is, despite widespread appearance of the abbreviation, `MACQL`, and its permutations. Consider Table 1, which shows how the same baseline method is purportedly included in each of the papers in our case study, yet the details provided for this algorithm are sparse, often lacking publicly available code, and can vary dramatically across papers.

To highlight these discrepancies and their effects more clearly, we note that the authors of each paper compare their proposed method with "`MACQL`" and claim superior performance. But *which* `MACQL` is being compared, has a significant bearing on the degree to which these conclusions are likely to hold. For example, consider the following simple experiment. We compare two viable candidates for `MACQL` across three SMACv1 (Samvelyan et al., 2019) maps (`8m`, `2s3z`, `5m v 6m`). Specifically, we compare MADDPG (Lowe et al., 2017) with the Gumbel-Softmax (Jang et al., 2016), against QMIX (Rashid et al., 2018), each with the addition of CQL. Importantly, note that in both cases, we could present such an algorithm as "`MACQL`" as used in prior work. However, the results in Figure 2 clearly reveal their relative differences in performance. Here we report the median, mean and interquartile mean (IQM) as recommended by Agarwal et al. (2022). We notice, too, that the outcome changes depending on the scenario used—MADDPG with CQL outperforms QMIX on `8m`, whereas the ordering is reversed on `5m_vs_6m`.

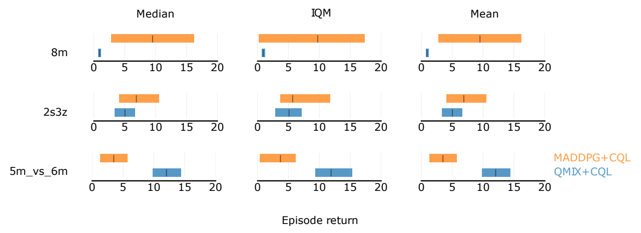


Figure 2: Comparing the performance of QMIX+CQL and MADDPG+CQL, two algorithms that could reasonably be called `MACQL` in the literature (see Table 1), using the Medium dataset from three different SMACv1 scenarios. We see that the difference in performance of these algorithms is significant, and depends on the scenario considered.

Mismatches in Scenarios Used for Comparison In addition to the ambiguity in baselines used for comparison, there is also a confusing mismatch in the selected scenarios used across prior work. Quite

Table 2: Depiction of which SMACv1 scenarios were used in the five papers from our case study (✓ present, ● absent).

Paper	`2s3z`	`3s v 5z`	`5m v 6m`	`6h v 8z`	`MMM`	`10m v 11m`	`3s v 3z`	`3s5z`	`1c3s5z`	`2c v 64zg`	`3s5z v 3s6z`	`corridor`
Yang et al. (2021)	✓	●	●	●	✓	✓	✓	●	●	●	●	●
Pan et al. (2022)	✓	●	●	●	●	●	●	✓	✓	✓	●	●
Meng et al. (2023)	✓	●	●	●	●	●	●	✓	●	●	✓	✓
Shao et al. (2023)	✓	✓	✓	✓	●	●	●	●	●	●	●	●
Wang et al. (2023)	●	●	✓	✓	●	●	●	●	●	✓	●	✓

Table 3: Summary of the evaluation methodologies from the five papers in our case study.

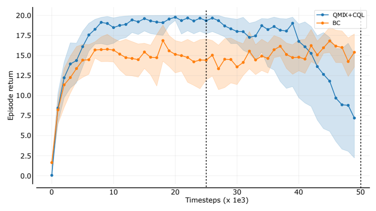
Paper	Evaluation frequency	Performance metrics	Results given as	Seeds
Yang et al. (2021)	10 episodes per 50 training episodes	Episode return	Plots	5
Pan et al. (2022)	Not reported	Normalised score	Plots	5
Meng et al. (2023)	32 evaluation epochs at points during training	Episode return	Plots	Not reported
Shao et al. (2023)	Not reported	Episode return, normalised score	Tabulated Values	5
Wang et al. (2023)	32 episodes, sourcing method not reported	Episode return	Tabulated Values	5

simply, consider Table 2, which shows the choice of scenarios from SMACv1 used for comparison across the different papers in our case study. There is not a single scenario that is used consistently in all papers. Furthermore, we find several instances where a scenario is unique to a specific paper, even though in practice, it should be trivial to ensure an overlap with all scenarios from prior work. As a result, it becomes difficult to trust and corroborate the results across different papers, and make meaningful comparisons when a new algorithm is proposed.

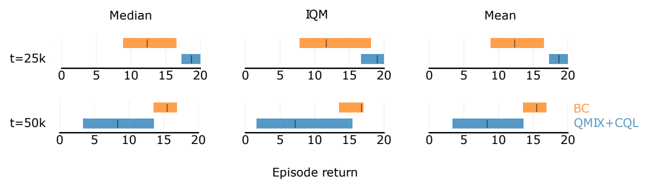
Inconsistencies in Evaluation Methodology Finally, we find several discrepancies in the way authors evaluate and present their results. Consider Table 3 which provides a summary of the approaches taken. Not only are there no dimensions along which evaluation is consistent, there are also gaps in the reporting of evaluation procedures, which make it difficult to compare results across studies. Furthermore, compared to the 10 seed standard recommended by Agarwal et al. (2022) and adopted by the online MARL community (Gorsane et al., 2022), using 5 seeds is most common in the papers from our case study. Owing to these small sample sizes the statistical validity of results could be questioned. However, often even this approximate measure of statistical uncertainty is ignored, where claims are made only based on which algorithms achieve the highest mean return across tasks. Consult the appendices for a visual comparison of the tabulated data for state-of-the-art (SOTA) claims. For many tasks the increase in reported performance is not statistically significant due to overlapping error bounds.

Notably, papers lack consideration for the effect that the computational training and online tuning budget can have on the outcome of reported experiments. Computational budget is especially important. In the online setting, environment interaction is often the most expensive part of training. However, in the offline setting, the greatest computational cost is likely updating the model parameters, and therefore for newly proposed algorithms this should be of great interest. Yet, hardware and time taken to train are typically not reported in papers. As for the online tuning budget, in the single-agent case Kurenkov and Kolesnikov (2022) note that this budget can have a significant impact on which offline RL algorithms are preferred across a variety of domains. We assert that this budget is equally important for the multi-agent case, but is not being controlled for by any paper in our case study.

To emphasise the importance of these two budgets, consider Figure 3, where we train two algorithms on the `8m` SMACv1 scenario. Notice how stopping the training at $25k$ steps yield very different conclusions to stopping at $50k$ steps. The training budget significantly affects the preference between algorithms. However, this can be accounted for using regular online evaluations - for example, if we would like QMIX+CQL to be preferred, we could stop training at $25k$ steps. For online MARL, this is acceptable since access to the environment has no limitations. For the use cases of offline MARL, however, we do not necessarily have access to regular online evaluations. So although we can observe a preferable stopping point, a fair evaluation of an offline algorithm would not be able to use earlystopping unless a large enough evaluation budget is specified. If an online tuning budget



(a) Sample efficiency with bootstrap confidence intervals (Agarwal et al., 2022).



(b) Performance reported at $25k$ steps and at $50k$ steps.

Figure 3: A comparison of the performance of behaviour cloning (BC) and QMIX+CQL on the SMACv1 `8m` scenario with the `Medium` dataset, across 10 seeds. Although QMIX+CQL outperforms BC during the first half of training, its performance deteriorates in the second half, making BC the preferred algorithm over the maximum training time.

is specified and training budget is considered an important hyperparameter, researchers can avoid unintentionally cherry-picking results.

The lack of consistency, transparency, and completeness of evaluation procedures in offline MARL slows progress by forcing researchers to perform expensive re-evaluations of baselines for their comparisons. But perhaps most damaging is the inability to compare and build upon prior work. This allows the field to maintain a mirage of steady progress, while in reality, algorithms are not becoming materially better.

3 Reevaluating the Empirical Evidence from Prior Work

Given the problems of baseline and evaluation methodology in offline MARL, we now revisit the empirical evidence from prior work through an independent standardised study.

Arguably, the gold standard for such a study would use the exact datasets from the authors and their original code (note, even this approach may have drawbacks such as potentially perpetuating poor experimental design and algorithm implementations). Alternatively, we could obtain a selection of similar yet distinct datasets (e.g. those from Formanek et al. (2023)), and use existing code implementations from the respective authors for each algorithm considered. Unfortunately, this approach often proves to be infeasible. In some works, only parts of the proposed algorithm code are shared (e.g. Pan et al. (2022) only share code for their method in continuous action spaces), and in many works, code for the baseline algorithms is omitted completely. As another alternative, we could decide to use our *own* algorithm implementations and datasets, however, this would put us in similar territory as prior work with regards to drawing concrete conclusions. As the most sensible middle ground, we do the following: we use the exact same datasets as provided in prior work, but train our own *baseline* implementations on these datasets; for the results of the other algorithms, we extract the values exactly as they are given in their respective papers. As an illustrative example, suppose we are comparing our implementation of MADDPG with CQL against the results from the OMIGA paper (Wang et al., 2023), in one of the scenarios from MAMuJoCo (Peng et al., 2021). Here, we take the *datasets* provided by Wang et al. (2023), train our MADDPG+CQL algorithm, and compare these results to the tabular values reported by Wang et al. (2023) themselves. We feel this methodology is the fairest to the original authors, especially in the situation where the publicly available code is lacking and/or broken. We also view the task of implementing our own baselines, instead of attempting to re-implement the author's proposed algorithm, as a more faithful exercise.

Nonetheless, this approach still has challenges, for it requires access to the datasets used by other authors. Regrettably, in some cases we could not access this data, either because it was never made publicly available, or because the provided download links were broken and multiple attempts to reach the original authors were unsuccessful. In Table 4, we summarise our dataset access record, across the papers in the case study.

The next challenge is that several works use modified versions of environments to generate their datasets, and to evaluate their algorithms. For example, Wang et al. (2023) modify the MAMuJoCo environment such that agents all receive a global observation of the environment, rather than decentralised, partial observations as is standard. They also use a different version of SMACv1, seemingly

Table 4: Summary of dataset accessibility and whether we benchmarked our baselines on them.

Paper	Environment	Number of Datasets	Accessibility	Benchmarked
Yang et al. (2021)	SMACv1	4	Link broken	●
Pan et al. (2022)	SMACv1	4	Not available	●
	MAMuJoCo	4	Yes	✓
	MPE	12	Yes	✓
Meng et al. (2023)	SMACv1	62	Download fails	●
Shao et al. (2023)	SMACv1	16	Yes	✓
	MAMuJoCo	4	Yes, from Pan et al. (2022)	✓
	MPE	12	Yes, from Pan et al. (2022)	✓
Wang et al. (2023)	SMACv1	12	Yes	✓
	MAMuJoCo	12	Yes	✓

first modified by Yu et al. (2022), that has important differences from the original. Similarly, Pan et al. (2022) include their own code for the Multi Particle Environments (MPE), which differs from the standardised and maintained version in PettingZoo (Terry et al., 2021). As a consequence, several datasets from different papers *seem* to share a common environment, but in reality do not (e.g., the `5m_vs_6m` datasets generated by Wang et al. (2023) are not compatible with the `5m_vs_6m` datasets from Shao et al. (2023)). To facilitate fair comparisons to each of the original works, we re-use the respective unique environment configuration, even when this is different from the standard. *We do not advocate this approach for future work* and note that standardisation is crucial going forward (which we discuss in more detail in the next section).

We use four baselines for our experiments—two for discrete action spaces, and two for continuous. In the discrete case, we implement BC, and independent Q-learners (IQL) (Tampuu et al., 2017) with CQL regularisation to stabilise offline training. Notably, these are very straightforward baselines that do not rely on any value factorisation or global state information. In continuous action spaces, we use independent DDPG agents with behaviour cloning regularisation—a naive multi-agent extension of the algorithm by Fujimoto and Gu (2021), which notably only requires a single line to change in our vanilla implementation of independent DDPG. Finally, we also test MADDPG (Lowe et al., 2017) with CQL to stabilise offline critic learning. Interestingly, this baseline is used in multiple works (Pan et al., 2022; Wang et al., 2023), but its reported performance is poor.

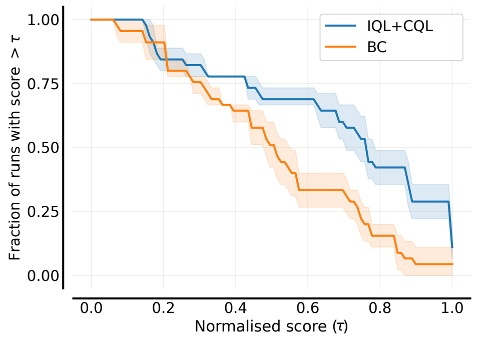
We train our baselines on MPE, SMACv1, and MAMuJoCo scenarios for $50k$, $100k$, and $200k$ training updates, respectively. At the end of training, we compute the mean episode return over 32 episodes and repeat each run across 10 independent random seeds. We avoid fine-tuning our algorithms on each scenario independently in an attempt to control for the online tuning budget (Kurenkov and Kolesnikov, 2022), and instead keep the hyperparameters fixed across each respective scenario.

Result We provide all the experimental results in tabular form in the appendix. From our own training results, we provide the mean episode return with the standard deviation across 10 independent seeds. As stated above, we extract values verbatim from prior work for other algorithms that are being compared. We perform a simple heteroscedastic, two-sided t-test with a 95% confidence interval for testing statistical significance, following Papoudakis et al. (2021). We summarise the tabulated results in an illustrative plot in Figure 1 (plotting our best baseline per dataset). To standardise comparisons, we normalise results from each dataset (i.e. scenario-quality-source combination) by the SOTA performance from the literature for that dataset. When our method is significantly better or equal to the best method in the literature, we indicate so using a star. We find that on 35 out of the 47 datasets tested, we match or surpass the performance of the current SOTA from the literature.[2]

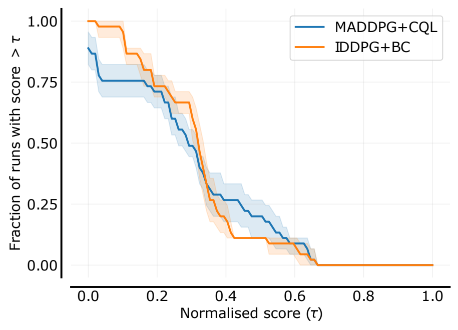
4 Standardising Baselines and Evaluation

The outcome from our benchmarking exercise paints a worrying picture of the state of offline MARL. We maintain that most of the contributions made by the research community to date are valuable.

[2]Code used to process results is available in a notebook: https://tinyurl.com/offline-marl-meta-review



(a) SMACv1



(b) MAMuJoCo

Figure 4: Performance profiles (Agarwal et al., 2022) aggregated across all results from Table 5 on SMACv1 and MAMuJoCo. Scores are normalised as per Fu et al. (2021).

However, because several works seem to be building upon unreliable baselines and using inconsistent evaluation protocols, the overall value to the community is diminished. We believe the community will be better served if we adopt a common set of datasets, baselines, and evaluation methodologies.

Datasets With regards to common datasets, OG-MARL (Formanek et al., 2023) includes a wide range of offline MARL datasets which the community has begun to adopt (Zhu et al., 2023; Yuan et al., 2023). We find that a notable advantage of OG-MARL datasets, aside from their ease of accessibility, is that they are generated on the standard environment configurations rather than customised ones. This significantly eases the challenge of matching datasets to environments, as highlighted in Section 3. Having said that, we also believe there is value in improving access to the datasets from prior works, which we used here for benchmarking. Thus, we convert all of the datasets available to us from the literature (see Table 4) to the OG-MARL dataset API to make them more easily available to the community, in one place. We include statistical profiles of the datasets in the appendix and give credit to the original creators.

Baselines While notable progress has been made standardising offline MARL datasets, we maintain that inconsistent use of baselines remains an overlooked issue in the field. Indeed, to highlight this, we conducted extensive benchmarking in Section 3. Now, to address the issue, we release our implementations of BC, IQL+CQL, IDDPG+BC, and MADDPG+CQL, as high-quality baselines for future research. Our baselines come with three main advantages. First, their demonstrated correctness, where we have shown they perform as well as, or better than, most algorithms in the literature on a wide range of datasets. Second, our baselines are easy to parse while also being highly performant on hardware accelerators. The core algorithm logic of our baselines is contained within a single file, which makes it easy for researchers to read, understand, and modify. In addition, all of the training code can be `jit` (just-in-time) compiled to XLA, making it very fast on hardware accelerators. Furthermore, we use the hardware accelerated replay buffers from Flashbax (Toledo et al., 2023), delivering performance gains by speeding up the time to sample from datasets. The third advantage is their compatibility with OG-MARL datasets, which comes "out of the box", offering the widest compatibility with offline MARL datasets in the literature [3]. As a foundation for future work, we provide tabular performance values across multiple scenarios for these algorithms, in Table 5. Furthermore, all raw training results can be viewed and downloaded, and are linked to in the appendix.

Evaluation Recent efforts have been made to address the issue in the online setting (Gorsane et al., 2022; Agarwal et al., 2022). However, similar efforts are still absent in the offline setting, as discussed in Section 2. Following in the spirit of this work, we propose a set of evaluation guidelines for offline MARL, given in the blue box on Page 8, which we believe will significantly improve research outcomes, if adopted.

[3]Datasets and baselines can be accessed at `https://github.com/instadeepai/og-marl`

Evaluation Guidelines for Cooperative Offline MARL

Choosing the settings to evaluate on:

- Select at least 2-3 different environments on which to test. For example, evaluating on both SMAC and MAMuJoCo is the most common combination. We encourage additionally evaluating on environments beyond these two, to avoid overfitting to them. Do not use non-standard environment configurations without explicitly stating so.
- For each environment choose at least 3-4 different scenarios. If the environment creators provide a minimal set of recommended scenarios, use those. Alternatively, focus on scenarios that are common in prior literature.
- Choose a range of dataset quality types. We recommend at least including a "good" dataset where the majority of samples are from good policies and a "mixed" dataset where samples come from a wide range of policies including good, medium and poor ones.
- Try to use common existing datasets from the literature (Formanek et al., 2023). If you include your own dataset, provide a clear reason for why, and make it easily accessible to the community.

Choosing the baselines to compare to:

- Choose at least 3-4 relevant baselines to compare to.
- Usually include behaviour cloning, especially on good datasets where it can be challenging to beat.
- Try to use common and existing implementations of baselines from the literature.
- If you include your own novel baseline, make the code available and easy to run for future comparisons. It is not sufficient to only share the code for the novel algorithm being proposed.

Choosing the training and evaluation parameters:

- For each environment, set a training budget and keep it constant across algorithms. For example, we used 100k training updates on SMAC and 200k on MAMuJoCo.
- If possible, do regular evaluations during training so that you can plot a training curve for analysis at the end of training. Do not use information from these evaluations to influence a training run online, as this would violate the assumptions around the training being offline.
- We recommend at every evaluation step unrolling for 32 episodes and reporting the average episode return.
- As per Agarwal et al. (2022), you should repeat each run across 10 random seeds.

Reporting your results:

- Report the final evaluation result, averaged across all 10 seeds, along with the standard deviation.
- If doing regular evaluations during training, also report the average and maximum episode return during training as per Papoudakis et al. (2021) and plot sample efficiency curves as per Gorsane et al. (2022).
- Use appropriate statistical significance calculations to report on whether your algorithms significantly outperform the baselines (Papoudakis et al., 2021; Agarwal et al., 2022).
- In addition to reporting results on a per-dataset basis, also report aggregated results across scenarios and dataset types. Results can be aggregated by first normalising them as per Fu et al. (2021) and then applying the utilities from *MARL-eval* (Gorsane et al., 2022) (e.g. performance profile plots, see Figure 4).

Table 5: Three return metrics—the Final, ⟨Maximum⟩, (Average)—from the two baseline algorithms in two respective environments, shown across various scenario and dataset quality combinations. Each result is presented as the mean and standard deviation, over 10 seeds. For the Final return, boldface indicates the best performing algorithm, and an asterisk (*) indicates a metric is not significantly different from the best performing metric in that situation, based on a heteroscedastic, two-sided t-test with 5% significance.

(a) SMACv1

Scenario	Quality	BC	IQL+CQL
8m	Good	14.94 ± 1.58	**20.00 ± 0.00**
		⟨19.60 ± 0.90⟩	⟨20.00 ± 0.00⟩
		(16.18 ± 3.19)	(18.80 ± 2.79)
	Medium	10.65 ± 1.41	**19.10 ± 1.24**
		⟨11.32 ± 1.77⟩	⟨20.00 ± 0.00⟩
		(9.71 ± 1.93)	(18.94 ± 2.82)
	Poor	**5.35 ± 0.44**	4.85 ± 0.12*
		⟨5.59 ± 0.20⟩	⟨5.06 ± 0.31⟩
		(5.20 ± 0.55)	(4.81 ± 0.48)
2s3z	Good	18.18 ± 1.06*	**19.60 ± 0.92**
		⟨19.44 ± 1.30⟩	⟨20.03 ± 0.05⟩
		(17.44 ± 2.53)	(19.17 ± 2.64)
	Medium	13.14 ± 2.03	**15.79 ± 0.98**
		⟨13.99 ± 2.78⟩	⟨18.04 ± 1.38⟩
		(12.04 ± 1.91)	(16.00 ± 2.53)
	Poor	6.42 ± 1.39*	**7.85 ± 1.19**
		⟨8.16 ± 0.53⟩	⟨9.47 ± 0.64⟩
		(6.57 ± 1.18)	(8.50 ± 1.33)
5m_vs_6m	Good	15.67 ± 3.49*	**16.19 ± 1.64**
		⟨17.88 ± 1.19⟩	⟨17.87 ± 2.17⟩
		(15.31 ± 3.35)	(14.33 ± 3.88)
	Medium	10.92 ± 0.93	**15.78 ± 2.93**
		⟨14.22 ± 2.00⟩	⟨17.65 ± 1.76⟩
		(11.52 ± 2.73)	(13.22 ± 3.46)
	Poor	7.33 ± 0.55	**10.87 ± 2.63**
		⟨8.38 ± 1.51⟩	⟨12.21 ± 2.21⟩
		(7.13 ± 1.05)	(9.99 ± 2.29)

(b) MAMuJoCo

Scenario	Quality	IDDPG+BC	MADDPG+CQL
2x4 Ant	Good	485.11 ± 508.76	**1803.24 ± 546.31**
		⟨965.37 ± 11.74⟩	⟨1776.19 ± 770.78⟩
		(284.76 ± 378.79)	(1296.62 ± 727.39)
	Medium	890.66 ± 234.86*	**1052.75 ± 182.30**
		⟨973.34 ± 23.98⟩	⟨1067.96 ± 129.09⟩
		(724.09 ± 264.01)	(922.47 ± 181.97)
	Poor	**888.83 ± 110.97**	504.24 ± 125.56
		⟨992.16 ± 54.86⟩	⟨979.42 ± 10.95⟩
		(891.38 ± 108.36)	(479.40 ± 157.92)
4x2 Ant	Good	**266.57 ± 75.26**	103.36 ± 604.36*
		⟨978.47 ± 2.93⟩	⟨1191.14 ± 641.59⟩
		(401.17 ± 282.74)	(540.98 ± 628.38)
	Medium	969.96 ± 175.01*	**1394.98 ± 350.67**
		⟨1318.59 ± 194.35⟩	⟨1477.25 ± 141.49⟩
		(1109.58 ± 213.13)	(1302.99 ± 266.38)
	Poor	**838.84 ± 59.49**	−1463.67 ± 1627.83
		⟨975.24 ± 10.54⟩	⟨966.86 ± 8.75⟩
		(861.92 ± 85.00)	(−841.30 ± 1322.39)
2x3 HalfCheetah	Good	**5411.68 ± 501.33**	3876.05 ± 995.80
		⟨5733.50 ± 290.89⟩	⟨2988.31 ± 1658.94⟩
		(4765.36 ± 1433.66)	(1735.93 ± 1811.54)
	Medium	**2819.67 ± 106.05**	2330.55 ± 277.40
		⟨2955.68 ± 240.16⟩	⟨2489.85 ± 321.24⟩
		(2482.85 ± 511.73)	(2204.64 ± 451.13)
	Poor	**676.65 ± 59.43**	−35.92 ± 32.53
		⟨703.75 ± 26.57⟩	⟨−4.72 ± 4.15⟩
		(630.68 ± 107.89)	(−46.00 ± 77.23)

5 Conclusion

We conducted a thorough analysis of prior work in offline MARL and identified several significant methodological failures which we demonstrated are inhibiting progress in the field. Furthermore, we benchmarked simple baselines against several proposed SOTA algorithms and showed that our baselines outperform them in most cases. We used these insights to propose improving standards in evaluation with a simple protocol.

Limitations Our work highlights some important challenges faced by the field of offline MARL, but does not capture *all* such challenges. We hope our contributions will make it easier for authors to align and compare their future work, but we ultimately realise that our efforts require vetting by the community, to be tested and validated over time. We welcome such engagements, to collectively chart a path forward.

References

R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice, 2022. 3, 4, 5, 7, 8

P. Barde, J. Foerster, D. Nowrouzezahrai, and A. Zhang. A model-based solution to the offline multi-agent reinforcement learning coordination problem. *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 2024. 1

C. Formanek, A. Jeewa, J. Shock, and A. Pretorius. Off-the-grid marl: Datasets and baselines for offline multi-agent reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, page 2442–2444, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450394321. 5, 7, 8

J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021. 7, 8

S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021. 6, 12, 13

R. Gorsane, O. Mahjoub, R. de Kock, R. Dubb, S. Singh, and A. Pretorius. Towards a standardised performance evaluation protocol for cooperative marl, 2022. 4, 7, 8

M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps, 2017. 13

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2016. 3

J. Jiang and Z. Lu. Offline decentralized multi-agent reinforcement learning, 2021. 1

A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2020. 3, 12, 13

V. Kurenkov and S. Kolesnikov. Showing your offline reinforcement learning work: Online evaluation budget matters. *International Conference on Machine Learning*, 2022. 4, 6

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

R. Lowe, Y. I. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 2017. 3, 6, 12

X. Lyu, Y. Xiao, B. Daley, and C. Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 844–852, 2021. 3

L. Meng, M. Wen, C. Le, X. Li, D. Xing, W. Zhang, Y. Wen, H. Zhang, J. Wang, Y. Yang, and B. Xu. Offline pre-trained multi-agent decision transformer. *Machine Intelligence Research*, 20(2): 233–248, Mar. 2023. ISSN 2731-5398. 1, 2, 3, 4, 6, 20

L. Pan, L. Huang, T. Ma, and H. Xu. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International conference on machine learning*, pages 17221–17237. PMLR, 2022. 1, 2, 3, 4, 5, 6, 15, 17, 19, 21

G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019. 1

G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. 6, 8

B. Peng, T. Rashid, C. A. S. de Witt, P.-A. Kamienny, P. H. S. Torr, W. Böhmer, and S. Whiteson. Facmac: Factored multi-agent centralised policy gradients, 2021. 5

R. F. Prudencio, M. R. Maximo, and E. L. Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. 1

T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *International Conference on Machine Learning*, 2018. 3

M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019. 3

J. Shao, Y. Qu, C. Chen, H. Zhang, and X. Ji. Counterfactual conservative q learning for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 37, 2023. 1, 2, 3, 4, 6, 15, 17, 18, 19, 20, 21

A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017. 6

J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 2021. 6

Q. Tian, K. Kuang, F. Liu, and B. Wang. Learning from good trajectories in offline multi-agent reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37:11672–11680, 06 2023. doi: 10.1609/aaai.v37i10.26379. 1

E. Toledo, L. Midgley, D. Byrne, C. R. Tilbury, M. Macfarlane, C. Courtot, and A. Laterre. Flashbax: Streamlining experience replay buffers for reinforcement learning with jax, 2023. URL `https://github.com/instadeepai/flashbax/`. 7

X. Wang and X. Zhan. Offline multi-agent reinforcement learning with coupled value factorization. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 2781–2783, 2023. 15

X. Wang, H. Xu, Y. Zheng, and X. Zhan. Offline multi-agent reinforcement learning with implicit global-to-local value regularization. *Advances in Neural Information Processing Systems*, 37, 2023. 1, 2, 3, 4, 5, 6, 15, 16, 19, 20, 21

Y. Yang, X. Ma, C. Li, Z. Zheng, Q. Zhang, G. Huang, J. Yang, and Q. Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:10299–10312, 2021. 1, 2, 3, 4, 6

C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35: 24611–24624, 2022. 6

L. Yuan, Z. Zhang, L. Li, C. Guan, and Y. Yu. A survey of progress on cooperative multi-agent reinforcement learning in open environment. *arXiv preprint arXiv:2312.01058*, 2023. 7

Y. Zhong, J. G. Kuba, X. Feng, S. Hu, J. Ji, and Y. Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25:1–67, 2024. 1

Z. Zhu, M. Liu, L. Mao, B. Kang, M. Xu, Y. Yu, S. Ermon, and W. Zhang. Madiff: Offline multi-agent learning with diffusion models. *Arxiv Preprint*, 2023. 7

A Algorithm implementation details and hyperparameters

Machine Learning Reproducibility Checklist (Algorithms, Code, and Experiment Details)

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, algorithm, and/or model. [Yes]
   (b) A clear explanation of any assumptions. [Yes]
   (c) An analysis of the complexity (time, space, sample size) of any algorithm. [No] The algorithms presented are foundational baselines that draw on existing works from the literature.
2. For all shared code related to this work, check if you include:
   (a) Specification of dependencies. [Yes] We provide a *requirements.txt* file and detailed installation instructions. In addition, we provide a working Dockerfile for maximum portability.
   (b) Training code. [Yes] All systems can be run using a single script. Instructions are provided in the README.
   (c) Evaluation code. [Yes] All systems have inbuilt evaluation and results are logged to the terminal and *Weights and Biases*.
   (d) (Pre-)trained model(s). [No] Pre-trained models have not been saved because training a model on any of the scenarios takes between a few minutes and at most a couple hours on a Laptop GPU (RTX 3070).
   (e) README file includes table of results accompanied by precise command to run to produce those results. [Yes] We provide a notebook with a database of results and visualisations and easy-to-run code for reproducing all results.
3. For all reported experimental results, check if you include:
   (a) The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results. [Yes] See below.
   (b) The exact number of training and evaluation runs. [Yes] Each experiment was repeated across 10 random seeds. For evaluation we rolled out policies for 32 episodes and computed the mean episode return.
   (c) A clear definition of the specific measure or statistics used to report results. [Yes] We measured episode return in all cases.
   (d) A description of results with central tendency (e.g. mean) & variation (e.g. error bars). [Yes]
   (e) The average runtime for each result, or estimated energy cost. [Yes] Depending on the scenario, reproducing a single experimental run can take between 20min and 4 hours on a Laptop GPU (e.g. RTX 3070). The MPE experiments are the fastest, followed by SMAC experiments and finally MAMuJoCo experiments take the longest.
   (f) A description of the computing infrastructure used. [Yes] Individual runs can easily be reproduced on a Laptop GPU (e.g. RTX 3070), 8GB of RAM and 4 CPU cores. However, reproducing all experiments on a single GPU would take approximately 20 days. We had access to a compute cluster with 10, roughly equivalent, GPUs. This meant that generating all baseline results took about 2 days.

A.1 IDDPG+BC (Continuous)

Our implementation draws on the minimalistic offline RL algorithm proposed by Fujimoto and Gu (2021). In essence, you simply add a behaviour cloning term to the deterministic policy gradient (DPG) loss. In a continuous action space with deterministic policies, this can be achieved by a simple mean square error between the output of the policy and the given action sampled from the dataset. As per Fujimoto and Gu (2021), we normalise the DPG term so that its scale is similar to the BC term and then use a *behaviour cloning weight* hyperparameter to control the relative importance of the behaviour cloning term vs. the DPG term. The critic conditioned on the environment state and the agents individual action only. Policies conditioned on the decentralised observations only. We used shared parameters by always concatenating an agent-ID to observations.

A.2 MADDPG+CQL (Continuous)

Our implementation adds CQL (Kumar et al., 2020) to MADDPG (Lowe et al., 2017). In the original version of CQL they had stochastic policies (soft actor critic), and so, getting actions *near* the current

Table 6: Hyper parameters used for IDDPG+BC across all MAMuJoCo and MPE datasets. We found that the recommended *behaviour cloning weight* of 2.5 proposed by Fujimoto and Gu (2021) worked well across all scenarios.

Hyperparameter	**Value**
Critic first linear layer	128
Critic second linear layer	128
Policy linear layer	64
Policy GRU layer	64
Critic learning rate	1e-3
Policy learning rate	3e-4
Target update rate	0.005
Discount (gamma)	0.99
BC weight	2.5

policy was simply a matter of sampling the stochastic policy. Since we had deterministic policies we applied a small amount of Gaussian noise to our actions. The amount of noise is then controlled by a hyperparameter we called *CQL sigma*. The *CQL weight* parameter controls the relative importance of the CQL loss in the overall critic loss. While the critic condition on joint-actions and the environment state, policies conditioned on the decentralised observations only. We used shared parameters by always concatenating an agent-ID to observations.

Table 7: Hyper parameters used for MADDPG+CQL across all MAMuJoCo and MPE datasets. We found that MADDPG+CQL was sensitive to the value of *CQL sigma* and the optimal value depended on the MuJoCo scenario. For *Ant* scenarios, 0.1 was the best value, while for *Hopper* and *HalfChetah* scenarios the best values were 0.2 and 0.3 respectively. This dependence on the scenario makes sense since it is well known that CQL has a dependence on the action space of the scenario tested on Kumar et al. (2020). We found that tuning the *CQL weight* across scenarios could also slightly improve performance but the value 3 worked relatively well across all scenarios. Future works could explore using automatic CQL weight tuning, and stochastic policies (e.g. soft actor critic) to remove the CQL sigma hyper-parameter.

Hyperparameter	**Value**
Critic first linear layer	128
Critic second linear layer	128
Policy linear layer	64
Policy GRU layer	64
Critic learning rate	1e-3
Policy learning rate	3e-4
Target update rate	0.005
Discount (gamma)	0.99
Number of CQL actions	10
CQL weight	3
CQL sigma	0.1, 0.2, 0.3

A.3 IQL+CQL (Discrete)

Our implementation added CQL (Kumar et al., 2020) to independent Q-Learners. Our independent Q-learners use recurrent Q-networks (Hausknecht and Stone, 2017). For the CQL loss, we simply sample a *number of CQL actions* randomly from the joint-action space and "push" their Q-values down, while pushing up the Q-values for joint-actions in the dataset The *CQL weight* hyperparameter controls the relative importance of the CQL term in the Q-Learning loss. The Q-networks condition on the decentralised observations only. We used shared parameters by always concatenating an agent-ID to observations.

A.4 Behaviour cloning (Discrete)

In our behaviour cloning implementation for discrete action spaces, we train policy networks to match the actions in the dataset using a simple categorical crossentropy loss. We use recurrent policy

Table 8: Hyper parameters used for IQL+CQL across all SMAC datasets.

Hyperparameter	**Value**
First linear layer	64
GRU layer	64
Learning rate	3e-4
Target period	200
Discount (gamma)	0.99
Number of CQL actions	10
CQL weight	2

networks which condition on the decentralised observations only. We used shared parameters by always concatenating an agent-ID to observations.

Table 9: Hyper parameters used for BC across all SMAC datasets.

Hyperparameter	**Value**
First linear layer	64
GRU layer	64
Learning rate	1e-3
Discount (gamma)	0.99

B Meta-review: Visualising statistical significance in the literature

We processed results reported in tabular form by Pan et al. (2022), Wang et al. (2023), Wang and Zhan (2023), and Shao et al. (2023). As previously stated, we perform a simple heteroscedastic, two-sided t-test with a 95% confidence interval for testing statistical significance. If we accept the null hypothesis, it can be said that with a 95% confidence interval For this section of the appendix, we consider only the results reported, and do not include our own baselines.

We show that many of the results (which form a large part of the evidence for SOTA claims for most of the considered papers) do not indicate a significant difference between performance of the algorithm with the highest mean and the next-best performing algorithm. Each result in each plot which has a red circle around it is equivalent to SOTA within the table in which it is reported.

B.1 OMAC

Figure 5 illustrates the results presented in Table 4 in the paper by Wang and Zhan (2023). We represent SMACv1 results on a $0 - 20$ scale to better interpret results within the scoring range. Note the proposed algorithm is unmatched on one dataset only. Additionally, mabcq (not the proposed algorithm) is equivalent to SOTA on all but 2 of the 12 datasets.

B.2 OMIGA

Figure 6 illustrates the results presented in Table 1 in the paper by Wang et al. (2023). We represent SMACv1 results on a $0 - 20$ scale to better interpret results within the scoring range. MAMuJoCo results are unnormalised.

Note the proposed algorithm is unmatched on only 3 of the 24 datasets. Additionally, maicq (not the proposed algorithm) is equivalent to SOTA on 18 of the 24 datasets.

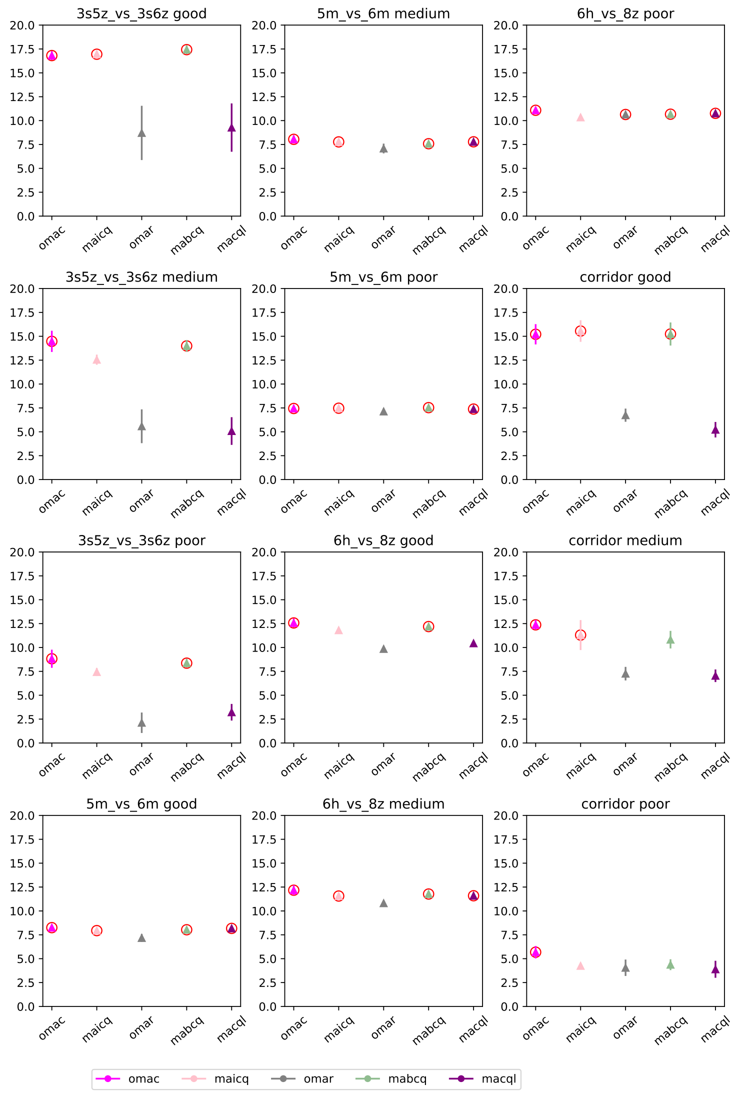


Figure 5: Results reported by Wang and Zhan (2023) on the SMAC environment.

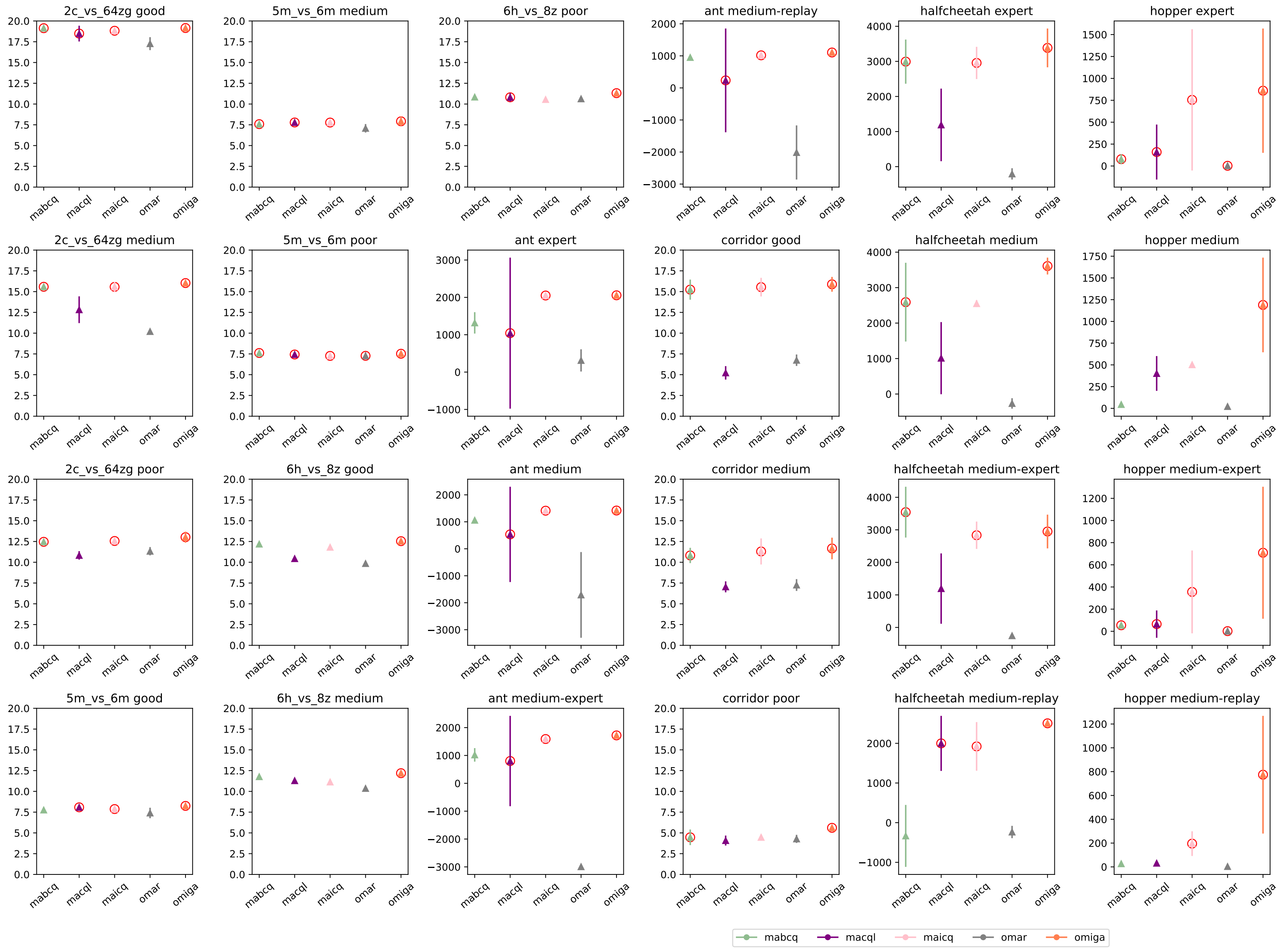


Figure 6: Results reported by Wang et al. (2023) on the SMAC and MAMuJoCo environments.

B.3 OMAR

Figure 7 illustrates the results presented in Table 1 in the paper by Pan et al. (2022). MPE results are normalised.

Note the proposed algorithm is unmatched on only 2 of the 12 datasets. Additionally, macql (not the proposed algorithm) is equivalent to SOTA on 10 of the 12 datasets.

B.4 CFCQL

Figure 7 illustrates the results presented in Table 1 in the paper by Shao et al. (2023). SMAC results are given in terms of win rate. MPE and MAMuJoCo results are normalised.

Note the proposed algorithm is unmatched on only 10 of the 32 datasets.

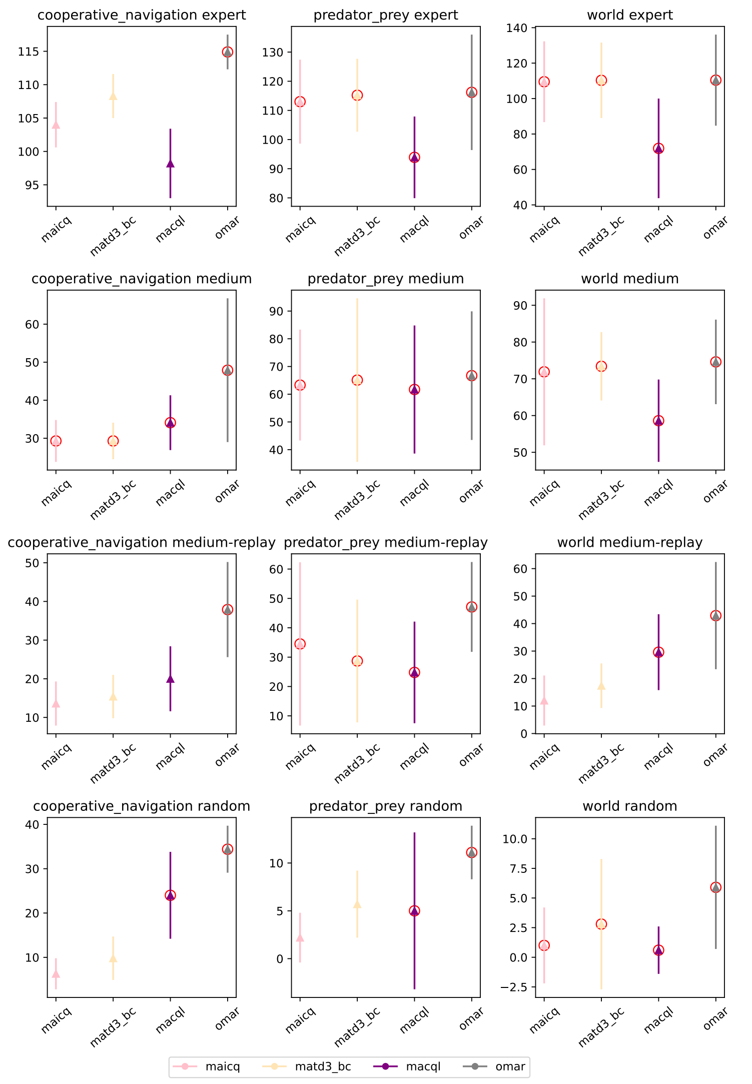


Figure 7: Results reported by Pan et al. (2022) on the MPE environment.

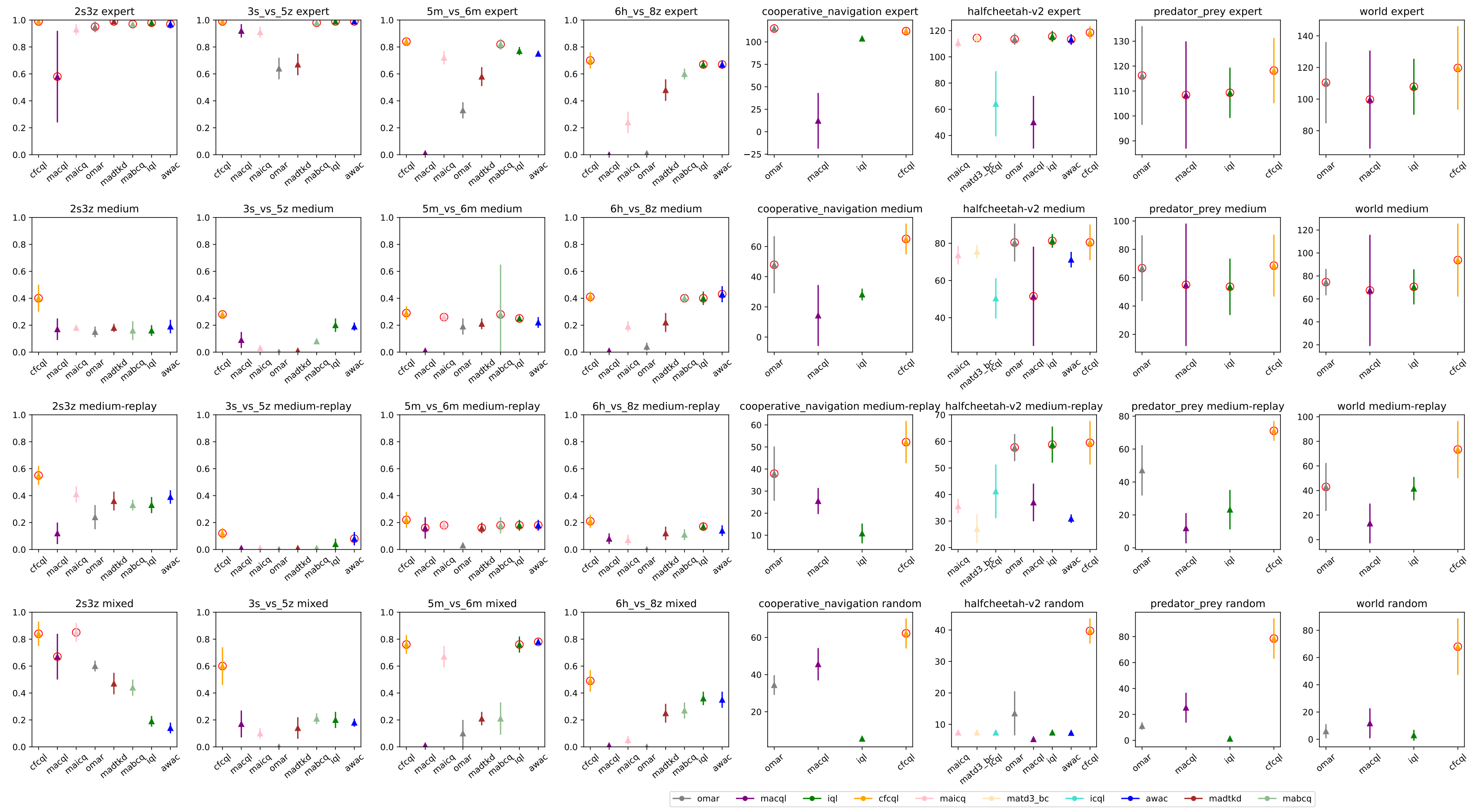


Figure 8: Results reported by Shao et al. (2023) in tabular form on SMAC, MPE and MAMuJoCo environments.

C Meta-review: Benchmarking literature results against our own baselines

The tabular results presented below are used to create Figure 1. Mean and standard deviation are provided - in our case, using 10 seeds, in the case of results from the literature, using 5. A bold result indicates that the mean is highest (one way to define SOTA) and a starred result indicates that the result is not statistically significantly worse or better than the bold result.

C.1 MPE

Results in the MPE environment are normalised. The given formula for normalisation is $100 \times (S - S_{random}) \div (S_{expert} - S_{random})$. We consider the cooperative navigation (CN), predator-prey (PP) and world (WD) scenario datasets. Both Shao et al. (2023) and Pan et al. (2022) provide results on these datasets. We were unfortunately unsuccessful in getting access to the PP and WD datasets. Furthermore, we faced challenges getting the customised environment working because it required loading pre-trained policies from the adversaries and there were limited details provided on how to do that.

Table 10: Mean and standard deviation results from Pan et al. (2022), Shao et al. (2023) and our baseline (IDDPG+BC) on MPE environment datasets provided by Pan et al. (2022).

		Pan et al.				Shao et al.			Our Baseline
		maicq	matd3 bc	macql	omar	macql	iql	cfcql	iddpg+bc
task	dataset quality								
CN	expert	104.00±3.40	108.30±3.30	98.20±5.20	**114.90±2.60**	12.20±31.00	103.70±2.50	112.00±4.00*	104.90±3.33
	medium	29.30±5.50	29.30±4.80	34.10±7.20	47.90±18.90	14.30±20.20	28.20±3.90	65.00±10.20	**102.02±10.68**
	medium-replay	13.60±5.70	15.40±5.60	20.00±8.40	37.90±12.30	25.50±5.90	10.80±4.50	52.20±9.60	**118.19±7.77**
	random	6.30±3.50	9.80±4.90	24.00±9.80	34.40±5.30	45.60±8.70	5.50±1.10	62.20±8.10	**130.33±15.68**
PP	expert	113.00±14.40*	115.20±12.50*	93.90±14.00	116.20±19.80*	108.40±21.50*	109.30±10.10*	**118.20±13.10**	Not available
	medium	63.30±20.00*	65.10±29.50*	61.70±23.10*	66.70±23.20*	55.00±43.20*	53.60±19.90*	**68.50±21.80**	Not available
	medium-replay	34.50±27.80	28.70±20.90	24.80±17.30	47.10±15.30	11.90±9.20	23.20±12.00	**71.10±6.00**	Not available
	random	2.20±2.60	5.70±3.50	5.00±8.20	11.10±2.80	25.20±11.50	1.30±1.60	**78.50±15.60**	Not available
WD	expert	109.50±22.80*	110.30±21.30*	71.90±28.10	110.40±25.70*	99.70±31.00*	107.80±17.70*	**119.70±26.40**	Not available
	medium	71.90±20.00*	73.40±9.30*	58.60±11.20*	74.60±11.50*	67.40±48.40*	70.50±15.30*	**93.80±31.80**	Not available
	medium-replay	12.00±9.10	17.40±8.10	29.60±13.80	42.90±19.50*	13.20±16.20	41.50±9.50	**73.40±23.20**	Not available
	random	1.00±3.20	2.80±5.50	0.60±2.00	5.90±5.20	11.70±11.00	2.90±4.00	**68.00±20.80**	Not available

C.2 MAMuJoCo

Shao et al. (2023) normalise the episode returns against the mean in the dataset.

Table 11: Results on MAMuJoCo datasets from OMAR.

		Shao et al.								Our Baselines	
		maicq	matd3 bc	icql	omar	macql	iql	awac	cfcql	iddpg+bc	maddpg+cql
task	dataset quality										
2x3 halfcheetah	expert	110.60±3.30	114.40±3.80	64.20±24.90	113.50±4.30	50.10±20.10	115.60±4.20*	113.30±4.10	118.50±4.90*	**120.02±1.80**	119.69±3.00*
	medium	73.60±5.00	75.50±3.70	50.40±10.80	80.40±10.20	51.50±26.70	81.30±3.70	71.20±4.20	80.50±9.60	156.86±4.53	**165.21±10.96**
	medium-replay	35.60±2.70	27.10±5.50	41.20±10.10	57.70±5.10	37.00±7.10	58.80±6.80	30.90±1.60	59.50±8.20	**719.03±37.31**	668.97±28.57
	random	7.40±0.00	7.40±0.00	7.40±0.00	13.50±7.00	5.30±0.50	7.40±0.00	7.30±0.00	**39.70±4.00**	Not available	Not available

In Wang et al. (2023) they give the raw episode returns.

Table 12: Results on MAMuJoCo datasets from OMIGA

		Wang et al.					Our Baselines	
		mabcq	macql	maicq	omar	omiga	iddpg+bc	maddpg+cql
task	dataset quality							
2x4 ant	expert	1317.73±286.28	1042.39±2021.65*	2050.00±11.86*	312.54±297.48	**2055.46±1.58**	1031.34±57.83	2053.35±12.02*
	medium	1059.60±91.22	533.90±1766.42*	1412.41±10.93*	-1710.04±1588.98	**1418.44±5.36**	365.73±54.77	1395.29±15.05
	medium-expert	1020.89±242.74	800.22±1621.52*	1590.18±85.61	-2992.80±6.95	1720.33±110.63	181.87±352.79	**1972.17±129.61**
	medium-replay	950.77±48.76	234.62±1618.28*	1016.68±53.51*	-2014.20±844.68	**1105.13±88.87**	859.27±66.68	951.95±10.50
3x1 hopper	expert	77.85±58.04	159.14±313.83	754.74±806.28	2.36±1.46	859.63±709.47	**3553.13±107.21**	82.49±103.59
	medium	44.58±20.62	401.27±199.88	501.79±14.03	21.34±24.90	1189.26±544.30*	795.35±29.69	**1745.88±607.83**
	medium-expert	54.31±23.66	64.82±123.31	355.44±373.86	1.44±0.86	709.00±595.66	**3087.47±721.82**	334.36±214.95
	medium-replay	26.53±24.04	31.37±15.16	195.39±103.61	3.30±3.22	774.18±494.27	231.40±167.45	**1738.38±291.67**
6x1 halfcheetah	expert	2992.71±629.65	1189.54±1034.49	2955.94±459.19	-206.73±161.12	3383.61±552.67	**4763.99±119.56**	-106.72±316.55
	medium	2590.47±1110.35*	1011.35±1016.94	2549.27±96.34	-265.68±146.98	**3608.13±237.37**	2739.01±51.09	3350.97±132.93*
	medium-expert	**3543.70±780.89**	1194.23±1081.06	2833.99±420.32*	-253.84±63.94	2948.46±518.89*	3400.65±454.49*	-183.98±629.66
	medium-replay	-333.64±780.89	1998.67±693.92	1922.42±612.87	-235.42±154.89	2504.70±83.47	**3380.58±121.08**	2795.40±282.13

C.3 SMAC

For our SMAC benchmark we managed to get datasets from Shao et al. (2023) and from Wang et al. (2023). We then implemented independent Q-learners with CQL. Since we used recurrent Q-networks we call our method IDRQN+CQL.

On the datasets from Wang et al. (2023), the episode returns were given.

Table 13: Results on SMAC datasets from Wang et al. (2023), subsampled from datasets generated by Meng et al. (2023).

				Wang et al.			Our Baseline
		mabcq	macql	maicq	omar	omiga	iql+cql
task	dataset quality						
2c vs 64zg	good	19.13±0.27	18.48±0.95*	18.82±0.17	17.27±0.78	19.15±0.32*	**19.52±0.26**
	medium	15.58±0.37*	12.82±1.61	15.57±0.61*	10.20±0.20	**16.03±0.19**	14.89±0.72
	poor	12.46±0.18*	10.83±0.51	12.56±0.18*	11.33±0.50	**13.02±0.66**	11.03±0.41
5m vs 6m	good	7.76±0.15	8.08±0.21	7.87±0.30	7.40±0.63	8.25±0.37	**12.36±1.09**
	medium	7.58±0.10	7.78±0.10	7.77±0.30	7.08±0.51	7.92±0.57	**12.30±0.74**
	poor	7.61±0.36	7.43±0.10	7.26±0.19	7.27±0.42	7.52±0.21	**10.20±0.75**
6h vs 8z	good	12.19±0.23	10.44±0.20	11.81±0.12	9.85±0.28	12.54±0.21*	**12.72±0.44**
	medium	11.77±0.16	11.29±0.29	11.13±0.33	10.36±0.16	**12.19±0.22**	12.01±0.42*
	poor	10.84±0.16	10.81±0.52*	10.55±0.10	10.63±0.25	**11.31±0.19**	10.41±0.36
corridor	good	15.24±1.21	5.22±0.81	15.54±1.12	6.74±0.69	15.88±0.89	**19.06±0.81**
	medium	10.82±0.92	7.04±0.66	11.30±1.57	7.26±0.71	11.66±1.30	**13.44±1.31**
	poor	4.47±0.94	4.08±0.60	4.47±0.33	4.28±0.49	5.61±0.35*	**6.11±1.10**

On the Shao et al. (2023) datasets, the win rates were given.

Table 14: Results on SMAC datasets from Shao et al. (2023).

					Shao et al.					Our Baseline
		cfcql	macql	maicq	omar	madtkd	mabcq	iql	awac	iql+cql
task	dataset quality									
2s3z	expert	**0.99±0.01**	0.58±0.34*	0.93±0.04	0.95±0.04*	**0.99±0.02**	0.97±0.02*	0.98±0.03*	0.97±0.03*	**0.99±0.03**
	medium	**0.40±0.10**	0.17±0.08	0.18±0.02	0.15±0.04	0.18±0.03	0.16±0.07	0.16±0.04	0.19±0.05	0.16±0.13
	medium-replay	**0.55±0.07**	0.12±0.08	0.41±0.06	0.24±0.09	0.36±0.07	0.33±0.04	0.33±0.06	0.39±0.05	0.33±0.08
	mixed	0.84±0.09	0.67±0.17	0.85±0.07	0.60±0.04	0.47±0.08	0.44±0.06	0.19±0.04	0.14±0.04	**0.97±0.03**
3s vs 5z	expert	**0.99±0.01**	0.92±0.05	0.91±0.04	0.64±0.08	0.67±0.08	0.98±0.02*	**0.99±0.01**	**0.99±0.02**	0.98±0.03*
	medium	**0.28±0.03**	0.09±0.06	0.03±0.01	0.00±0.00	0.01±0.01	0.08±0.02	0.20±0.05	0.19±0.03	0.13±0.07
	medium-replay	0.12±0.04	0.01±0.01	0.01±0.02	0.00±0.00	0.01±0.01	0.01±0.01	0.04±0.04	0.08±0.05	**0.26±0.16**
	mixed	0.60±0.14*	0.17±0.10	0.10±0.04	0.00±0.00	0.14±0.08	0.21±0.04	0.20±0.06	0.18±0.03	**0.67±0.18**
5m vs 6m	expert	**0.84±0.03**	0.01±0.01	0.72±0.05	0.33±0.06	0.58±0.07	0.82±0.04*	0.77±0.03	0.75±0.02	0.72±0.11
	medium	**0.29±0.05**	0.01±0.01	0.26±0.03*	0.19±0.06	0.21±0.04	0.28±0.37*	0.25±0.02*	0.22±0.04	0.19±0.09
	medium-replay	**0.22±0.06**	0.16±0.08*	0.18±0.04*	0.03±0.02	0.16±0.04*	0.18±0.06*	0.18±0.04*	0.18±0.04*	**0.22±0.16**
	mixed	0.76±0.07*	0.01±0.01	0.67±0.08	0.10±0.10	0.21±0.05	0.21±0.12	0.76±0.06*	**0.78±0.02**	0.65±0.16
6h vs 8z	expert	**0.70±0.06**	0.00±0.00	0.24±0.08	0.01±0.01	0.48±0.08	0.60±0.04	0.67±0.03*	0.67±0.03*	0.60±0.17*
	medium	0.41±0.04*	0.01±0.01	0.19±0.04	0.04±0.03	0.22±0.07	0.40±0.03*	0.40±0.05*	**0.43±0.06**	0.32±0.17*
	medium-replay	0.21±0.05*	0.08±0.04	0.07±0.04	0.00±0.00	0.12±0.05*	0.11±0.04	0.17±0.03*	0.14±0.04*	**0.22±0.13**
	mixed	0.49±0.08	0.01±0.01	0.05±0.03	0.00±0.00	0.25±0.07	0.27±0.06	0.36±0.05	0.35±0.06	**0.63±0.12**

D Converted Datasets

Machine Learning Reproducibility Checklist (Datasets)

1. For all datasets used, check if you include:
   - (a) The relevant statistics, such as number of examples. [Yes] See provided notebook and sample code below for analysing datasets.
   - (b) The details of train / validation / test splits. [No] Not applicable for offline MARL in this case.
   - (c) An explanation of any data that were excluded, and all pre-processing step. [Yes] A full explanation in the main body of the text and below; we converted the datasets to the vault API.
   - (d) A link to a downloadable version of the dataset or simulation environment. [Yes] See below.
   - (e) For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control. [No] Not applicable; all datasets used are from the literature.

To compare existing results to our own baselines, we converted datasets from the literature into the Vault format from Flashbax. Examples of our methodology for converting these datasets are given in a notebook here: `https://bit.ly/vault-conversion-notebook`. Table 15 provides links to all Vault datasets.

Table 15: Links to Vault datasets converted from the literature.

Paper	Environment	Scenario	Link
Pan et al. (2022)	MAMuJoCo	2halfcheetah	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omar/mamujoco/2halfcheetah.zip
	MPE	simple-spread	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omar/mpe/simple_spread.zip
		simple-tag	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omar/mpe/simple_tag.zip
		simple-world	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omar/mpe/simple_world.zip
Shao et al. (2023)	SMACv1	2s3z	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/cfcql/smac_v1/2s3z.zip
		3s_vs_5z	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/cfcql/smac_v1/3s_vs_5z.zip
		5m_vs_6m	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/cfcql/smac_v1/5m_vs_6m.zip
		6h_vs_8z	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/cfcql/smac_v1/6h_vs_8z.zip
Wang et al. (2023)	SMACv1	corridor	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omiga/smac_v1/corridor.zip
		2c_vs_64zg	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omiga/smac_v1/2c_vs_64zg.zip
		5m_vs_6m	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omiga/smac_v1/5m_vs_6m.zip
		6h_vs_8z	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omiga/smac_v1/6h_vs_8z.zip
	MAMuJoCo	2ant	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omiga/mamujoco/2ant.zip
		3hopper	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omiga/mamujoco/3hopper.zip
		6halfcheetah	https://huggingface.co/datasets/InstaDeepAI/og-marl/resolve/main/prior_work/omiga/mamujoco/6halfcheetah.zip

Statistical profiles The statistical profiles (histograms and violin plots) of the converted datasets are available in the respective folders on HuggingFace: `https://huggingface.co/datasets/InstaDeepAI/og-marl/tree/main/prior_work`. Additionally, summary statistic values can easily be accessed using the demonstrative notebook at: `https://github.com/instadeepai/og-marl/blob/main/examples/dataset_analysis_demo.ipynb`.