The Journal of Open Source Software

PyTupli: Enabling Collaboration in Offline Reinforcement Learning

Hannah Markgraf[1¶], Michael Eichelbeck[1], Daria Cappey[1], Selin Demirtürk[1], Yara Schattschneider[1], and Matthias Althoff[1]

1 Technical University of Munich, Germany ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 10 December 2025
Published: unpublished

Summary

Offline reinforcement learning (RL) offers a powerful way to derive effective decision-making policies for control problems using pre-collected data. However, managing and sharing such datasets in collaborative research settings can be challenging, as proper versioning, filtering, and access control are required. PyTupli is a free, Python-based toolkit designed to support this workflow by providing an easy-to-use yet specialized framework that remains independent of external service providers. Unlike centralized platforms, PyTupli is designed for self-hosted deployment, giving research teams full control over their data infrastructure and access policies. Its client library allows users to serialize and store control problems, upload new data, and retrieve precisely the subsets they need through flexible and expressive filters. Built-in metrics help evaluate dataset coverage and utility, informing both dataset selection and algorithm design for offline RL. To ensure secure deployment, a container-based server component offers authentication, role-based access control, and automated certificate provisioning. Together, these capabilities enable researchers to create, manage, exchange, and analyze datasets for offline RL in a robust and accessible manner.

Statement of need

Reinforcement learning (RL) provides powerful methods for decision-making under uncertainty, but training RL agents typically requires extensive interaction with the underlying system or a computationally expensive simulator. Offline RL has emerged as a paradigm that alleviates this requirement by training agents solely on previously collected data (Lange et al., 2012). Such datasets contain tuples consisting of *state*, *action*, *next state*, and *reward*, obtained from recordings of real systems or generated through simulators.

Effectively managing, sharing, and curating these datasets is essential for collaborative offline RL research, yet existing tools provide only partial support. Platforms like Zenodo or the free version of HuggingFace allow users to share finalized datasets with a general public but are not suitable for ongoing or private collaborations. Furthermore, they lack mechanisms for tracking internal dataset structure or performing efficient, fine-grained queries. The same applies to version control systems such as GitHub. Traditional databases are better suited for this purpose but require substantial expertise to design and maintain robust workflows.

PyTupli addresses this gap by providing a dedicated Python toolkit for creating, storing, and sharing tuple datasets for custom environments. We provide a Docker container that each research group or collaboration can deploy independently to maintain complete control over their data. Each dataset is associated to a benchmark. Benchmarks are stored as JSON-serialized objects and can be linked to related artifacts, including time-series data, algorithm hyperparameters, or trained policies, allowing multiple benchmarks to reference shared resources. Because it is built for scalable collaboration, PyTupli includes integrated user and access management features.

Since the performance of offline RL algorithms often depends critically on the quality of the underlying dataset (Asadulaev et al., 2025; Schweighofer et al., 2022; Suttle et al., 2025), PyTupli offers extensive filtering capabilities. Whereas established offline RL datasets primarily support filtering by entire episodes (Liu et al., 2023; Younis et al., 2024), PyTupli enables tuple-level filtering as well. This can be used, for example, to rebalance datasets with sparse rewards or selectively include transitions from specific regions of the state space. In addition, PyTupli provides a suite of metrics for assessing dataset coverage and reward characteristics, which can serve as predictors of offline RL performance (Asadulaev et al., 2025; Schweighofer et al., 2022) and help guide algorithm selection.

Related Software

Publicly available tuple datasets have been essential for advancing offline RL algorithms (Kostrikov et al., 2021; Kumar et al., 2020). These curated collections span various domains, such as robotics and games (Formanek et al., 2023; Fu et al., 2020; Gulcehre et al., 2020; Younis et al., 2024), power system control (Qin et al., 2022), and autonomous driving (Lee et al., 2024; Liu et al., 2023), and are typically designed to support the development of improved offline RL methods. As offline RL techniques mature, they are being applied to increasingly diverse and task-specific control problems. Yet, to the best of our knowledge, no existing toolbox supports the collaborative creation, management, and sharing of datasets for custom environments. Minari (Younis et al., 2024) is the closest related tool, offering a repository of standardized datasets along with functionality for filtering, environment reconstruction, and recording new interactions. Its focus, however, remains on distributing datasets for established benchmarks.

PyTupli instead targets collaborative workflows for custom control tasks. It enables researchers to share datasets and benchmarks directly within project teams without dependence on a central public server. Although Minari permits users to request publication on its official platform, this approach is often unsuitable for ongoing or proprietary work or for datasets that evolve over time. In addition, Minari does not provide tools for assessing dataset quality or coverage, which PyTupli includes to support informed dataset curation and algorithm selection in offline RL research.

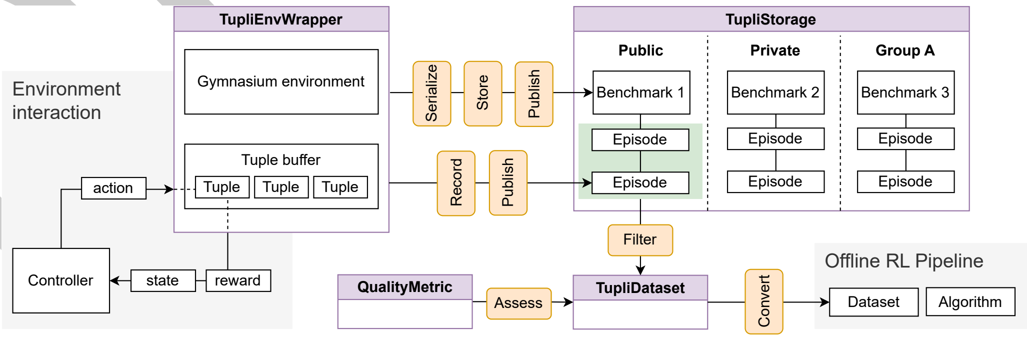

Figure 1: Overview of the core functionalities of PyTupli.

Core Functionalities

We briefly describe the core functionalities of PyTupli which are illustrated in Figure 1.

Benchmark and Artifact Management: PyTupli enables users to store any control task defined as a gymnasium environment. Environments may include parameterizable configurations that produce variations in task dynamics. A fully specified environment is stored as a benchmark, providing a unique reference for reproducible evaluation of controllers. Benchmarks can reference additional data, such as exogenous inputs, time-series data, or pre-trained models. These external units, referred to as artifacts, are stored independently and linked to multiple benchmarks to avoid duplication.

Data Management: PyTupli supports ingesting, storing, and querying structured datasets (RL tuples), including their relation to existing benchmark problems and any relevant metadata. MongoDB serves as the backend, providing scalable storage and efficient retrieval for large datasets. Table 1 shows how ingestion and retrieval times scale with dataset size.

Multi-User Collaboration and Access Control: PyTupli facilitates collaborative workflows through private, group, and public scopes. Based on their assigned role, users can store, retrieve, delete, and publish objects. A server-side backend with FastAPI provides a REST interface for secure, programmatic access, while token-based authentication ensures secure sharing across teams or organizations.

Integration with Existing Offline RL Infrastructure: An interface to the gymnasium framework enables users to record interactions with gymnasium environments as RL tuples. Furthermore, retrieved tuple datasets are made available in a form that can easily be converted into the dataset formats used by existing offline RL libraries such as d3rlpy (Seno & Imai, 2022).

Assessment of Dataset Quality: PyTupli implements metrics to evaluate dataset quality in terms of coverage and expected returns. These metrics inform dataset selection and can provide guidance for algorithm choice. Detailed formulations are provided in the following section.

Table 1: Upload and download times for established datasets averaged over 10 runs. We chose two examples with low, medium, and high dataset size from the Minari collection. However, not only the size, but also the nature of observations has a strong influence on processing times.

Dataset	Size	M	N	Upload (s)	Download (s)
D4RL					
door/human-v2	3.5MB	25	7K	0.83	0.24
hammer/human-v2	6.2MB	25	11K	1.11	0.40
antmaze/medium-play-v1	605.2MB	1K	1M	173.26	126.62
Atari					
pitfall/expert-v0	351.7MB	10	65K	18.56	16.51
Mujoco					
ant/expert-v0	1.92GB	2K	2M	64.17	29.65
humanoid/expert-v0	2.95GB	1K	999K	96.61	55.32

Quality Metrics

Return-Based Metrics

Return-based metrics, such as trajectory quality (TQ) (Schweighofer et al., 2022) or the average Q-value (Asadulaev et al., 2025) can inform algorithm decision. For example, Schweighofer et al. (2022) show that behavioral cloning performs well despite its simplicity if the dataset has a high TQ. For datasets with low TQ, algorithms from the deep Q-network family perform well in their experiments as they do not constrain the learned policy towards the distribution of the behavioral policy. TQ normalizes the average return of a dataset with respect to the returns obtained by a minimal performant and an expert policy. To provide similar insights without relying on such additional information, estimated relative return improvement (Swazinna et al., 2021) relates the maximum trajectory return in the dataset to its average return. While estimated return improvement and TQ operate on a trajectory level, average

The Journal of Open Source Software

Q-value estimation offers insights on the tuple level, making it a better predictor of offline RL performance (Asadulaev et al., 2025). It requires fitting a Q-function using Bellman updates, which is closely related to the objectives used in offline RL training. However, for continuous action spaces, the user needs to provide an evaluation policy to predict the next actions in the Bellman target. Such a policy can, for example, be obtained using behavioral cloning on the dataset.

Coverage-Based Metrics

An important question when assessing a dataset is whether the behavioral policy (or policies) used to generate it did explore the state and action space well enough to learn a meaningful target policy from the data. A common approach for quantifying explorativeness is to approximate the entropy of transition probabilities for the behavior policy. For discrete state and action spaces, Schweighofer et al. (2022) suggest to approximate this by counting unique state-action pairs. Optionally, this value can be normalized using a reference dataset $\mathcal{D}_{\text{ref}}$ of same size, for example, the replay buffer collected during online training. Schweighofer et al. (2022) show that low state-action coverage values hinder performance of a large variety of algorithms. While counting unique state-action pairs aims at estimating the Shannon entropy of the transition probabilities, Suttle et al. (2025) suggest that datasets that maximize their proposed behavioral entropy metric support better offline RL performance. They suggest a k-nearest-neighbor estimator of the true behavioral entropy that relies on density-based weighting of different regions in the state-action space.

Acknowledgements

This work was partially supported by the German Research Foundation (AL 1185/9-1).

References

Asadulaev, A., Karray, F., & Takac, M. (2025). Expert or not? Assessing data quality in offline reinforcement learning. *arXiv Preprint arXiv:2510.12638*. https://doi.org/10.48550/arXiv.2510.12638

Formanek, C., Jeewa, A., Shock, J., & Pretorius, A. (2023). *Off-the-grid MARL: Datasets and baselines for offline multi-agent reinforcement learning*. 2442–2444. https://doi.org/10.5555/3545946.3598961

Fu, J., Kumar, A., Nachum, O., Tucker, G., & Levine, S. (2020). D4rl: Datasets for deep data-driven reinforcement learning. *arXiv Preprint arXiv:2004.07219*. https://doi.org/https://doi.org/10.48550/arXiv.2004.07219

Gulcehre, C., Wang, Z., Novikov, A., Paine, T., Gómez, S., Zolna, K., Agarwal, R., Merel, J. S., Mankowitz, D. J., Paduraru, C., & others. (2020). Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, *33*, 7248–7259. https://doi.org/10.5555/3495724.3496332

Kostrikov, I., Nair, A., & Levine, S. (2021). Offline reinforcement learning with implicit q-learning. *arXiv Preprint arXiv:2110.06169*. https://doi.org/10.48550/arXiv.2110.06169

Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020). Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, *33*, 1179–1191. https://doi.org/10.5555/3495724.3495824

Lange, S., Gabel, T., & Riedmiller, M. (2012). Batch reinforcement learning. In *Reinforcement learning: State-of-the-art* (pp. 45–73). Springer. https://doi.org/10.1007/978-3-642-27645-3_2

The Journal of Open Source Software

Lee, D., Eom, C., & Kwon, M. (2024). AD4RL: Autonomous driving benchmarks for offline reinforcement learning with value-based dataset. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 8239–8245. https://doi.org/10.1109/ICRA57147.2024.10610308

Liu, Z., Guo, Z., Lin, H., Yao, Y., Zhu, J., Cen, Z., Hu, H., Yu, W., Zhang, T., Tan, J., & others. (2023). Datasets and benchmarks for offline safe reinforcement learning. *arXiv Preprint arXiv:2306.09303*. https://doi.org/10.48550/arXiv.2306.09303

Qin, R.-J., Zhang, X., Gao, S., Chen, X.-H., Li, Z., Zhang, W., & Yu, Y. (2022). NeoRL: A near real-world benchmark for offline reinforcement learning. *Advances in Neural Information Processing Systems*, *35*, 24753–24765. https://doi.org/10.5555/3600270.3602065

Schweighofer, K., Dinu, M., Radler, A., Hofmarcher, M., Patil, V. P., Bitto-Nemling, A., Eghbal-Zadeh, H., & Hochreiter, S. (2022). A dataset perspective on offline reinforcement learning. *Conference on Lifelong Learning Agents*, 470–517. https://doi.org/10.48550/arXiv.2111.04714

Seno, T., & Imai, M. (2022). d3rlpy: An offline deep reinforcement learning library. *Journal of Machine Learning Research*, *23*(315), 1–20. https://doi.org/10.5555/3586589.3586904

Suttle, W. A., Suresh, A., & Nieto-Granda, C. (2025). Behavioral entropy-guided dataset generation for offline reinforcement learning. *arXiv Preprint arXiv:2502.04141*. https://doi.org/10.48550/arXiv.2502.04141

Swazinna, P., Udluft, S., & Runkler, T. (2021). Measuring data quality for dataset selection in offline reinforcement learning. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–8. https://doi.org/10.1109/SSCI50451.2021.9660006

Younis, O. G., Perez-Vicente, R., Balis, J. U., Dudley, W., Davey, A., & Terry, J. K. (2024). *Minari* (Version 0.5.0). Zenodo. https://doi.org/10.5281/zenodo.13767625