The Journal of Open Source Software

Mighty: A Comprehensive Tool for studying Generalization, Meta-RL and AutoRL

Aditya Mohan [1*¶], **Theresa Eimer** [1*], **Carolin Benjamins** [1], **Marius Lindauer** [1,3], **and Andre Biedenkapp** [2]

1 Leibniz University Hannover, Germany **2** University of Freiburg, Germany **3** L3S Research Center, Germany ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 20 January 2026
Published: unpublished

Summary

Robust generalization, rapid adaptation, and automated tuning are critical for deploying reinforcement learning (RL) in real-world settings. Yet research in these areas remains fragmented across non-standard codebases and custom orchestration scripts. We introduce *Mighty*, an open-source library that unifies contextual generalization, Meta-RL, and AutoRL within a single modular interface. Mighty cleanly separates the *Agent* from a configurable environment modeled as a *Contextual MDP*, decoupling *inner-loop* updates from *outer-loop* adaptations. This enables unified support for: (i) contextual generalization and curriculum learning (e.g., unsupervised environment design), (ii) bi-level meta-learning (e.g., MAML, black-box strategies), and (iii) automated hyperparameter and architecture search (e.g. Bayesian optimization, evolutionary strategies, population-based training). We outline Mighty's design and validate its implementation on standard RL benchmarks. By offering a unified modular platform, Mighty simplifies experimentation and accelerates research on robust, adaptable RL.

Statement of need

Reinforcement learning (RL) has emerged as a powerful decision-making paradigm in complex and dynamic environments. Despite impressive successes in domains such as games (Badia et al., 2020; Silver et al., 2016; Vasco et al., 2024) and robotics (Lee et al., 2020), RL algorithms frequently overfit their training conditions and struggle to generalize to new tasks (Benjamins et al., 2023; Kirk et al., 2023; Mohan et al., 2024). Addressing this challenge requires methods that not only learn efficiently on a single task but also adapt rapidly to novel settings and automatically tune their learning process.

Recent research has advanced in three complementary directions: (i) Generalization in RL (Benjamins et al., 2023; Cho et al., 2024; Mohan et al., 2024), (ii) Meta-RL methods (Beck et al., 2023; Kaushik et al., 2020), and (iii) Automated RL (AutoRL) (Eimer et al., 2023; Mohan et al., 2023; Parker-Holder et al., 2022). Although each has led to promising algorithms, researchers frequently resort to fragmented codebases and ad hoc scripting across environment design, RL training, and meta-optimization. This fragmentation increases engineering effort, impedes rapid iteration, and undermines reproducibility (Dizon-Paradis et al., 2024).

We introduce *Mighty*: a modular library designed to enable research at the intersection of generalization, Meta-RL, and AutoRL. Mighty enforces a clean and principled separation between inner- and outer-loop processes, making it easy to combine, for example, curricula, context adaptation, and automated tuning within a unified framework. Users can prototype new methods, compose existing ones, and run controlled comparisons - all without ad hoc orchestration code.

The Journal of Open Source Software

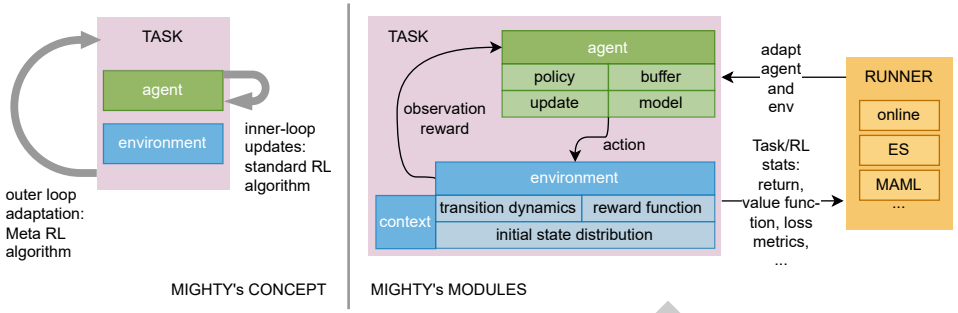

Figure 1: Overview of Mighty's concept and modules.

Mighty is designed around three design principles: *flexibility, smooth integration with existing libraries, and environment parallelization*. First, flexibility is central. Mighty exposes transitions, predictions, networks, and environments to meta-methods, enabling a broad range of research patterns including black-box outer loops, algorithm-informed inner loops, and environment-level interventions. Second, Mighty integrates smoothly with Gymnasium (Towers et al., 2024), Pufferlib (Suarez, 2025), CARL (Benjamins et al., 2023), and can interface with tools such as evosax (Lange, 2022) in under 100 lines of code. This minimizes the glue code while preserving flexibility. Finally, Mighty uses standard Python and PyTorch for optimized networks with vectorized CPU environments for fast environment interaction. This design offers high training speeds, even for purely CPU-based environments, without sacrificing algorithmic modularity or code clarity.

Existing Tools for RL and Meta RL

The rapidly growing ecosystem of RL libraries spans diverse design philosophies – from low-level composability (Weng et al., 2022) to turnkey baselines (Huang et al., 2022; Raffin et al., 2021) and massive-scale engines (Toledo, 2024) – making direct comparison and tool selection challenging. Modular research frameworks expose the internal building blocks of an RL pipeline as standalone components that can be re-combined to quickly prototype new algorithms. TorchRL (Bou et al., 2023) pioneered this approach in the PyTorch ecosystem, introducing the TensorDict abstraction to seamlessly pass the observations, actions and rewards between modules. Tianshou (Weng et al., 2022) offers a similarly flexible design with separate *Policy*, *Collector*, and *Buffer* classes, enabling researchers to switch custom exploration strategies or data collection schemes with minimal boilerplate. Although these libraries excel at inner loop algorithm development and fine-grained experimentation, counter to Mighty, they leave higher-order workflows such as curriculum learning or meta-adaptation across tasks to external scripts or user-written loops. Monolithic baselines such as stable baselines3 (SB3) (Raffin et al., 2021) and CleanRL/PureJaxRL (Huang et al., 2022; Lu et al., 2022) prioritize ease of use and reproducibility. However, this simplicity comes at the cost of extensibility: SB3's algorithms hide most of the training loop behind a single `learn()` call, and CleanRL's single file scripts are not designed for import or extension. Scalable platforms such as RLlib (Liang et al., 2018; Wu et al., 2021) and STOIX (Toledo, 2024) focus on maximizing throughput and supporting distributed execution. Although these systems shine when running large experiments, their APIs do not natively unify component modularity with built-in meta-learning or curriculum design. Mighty occupies the middle ground, offering efficient single-node performance via PyTorch, straightforward multicore environment parallelism, and a modular interface within the same cohesive framework.

The Journal of Open Source Software

Key Features

Mighty accelerates development and experimentation through an intuitive interface, modular algorithms, and flexible support for meta-methods extending beyond vanilla RL.

User Interface: Mighty prioritizes usability and flexibility. We use Hydra (Yadan, 2019) for structured configuration files that expose all relevant training details without overwhelming new users. This also plugs Mighty into Hydra's ecosystem for cluster execution and hyperparameter optimization. The algorithm components in Mighty are modular and can be replaced via configurations, allowing users to integrate new components without editing the training loop. *This keeps projects small, maintainable, and research-focused*. For example, to integrate domain randomization (Tobin et al., 2017) via Syllabus (Sullivan et al., 2025), we need around 100 lines of code each to interface Syllabus and build a custom task wrapper. With the Mighty project template as a base, *less than* 200 *lines of Python code and three configuration files* are enough for a full evaluation, including hyperparameter optimization and cluster deployment (see the project repository including results).

Agent Framework: Mighty includes three base RL algorithms – DQN (Mnih et al., 2015), SAC (Haarnoja et al., 2018) and PPO (Schulman et al., 2017) – built from four modular components: exploration policy, replay buffer, update function, and model parameterization. Each component is easily extendable, allowing users to swap in new methods without rewriting the entire algorithm or touching the training loop. Since these modules capture most of the algorithmic logic, this design supports a wide range of research. Our documentation features an overview on when and how to use each of Mighty's abstractions.

Meta-Learning Framework: Mighty's support for meta-methods is unique in the RL landscape. It offers two key abstractions: *runners* and *meta-components*. Runners control training lifecycles, interacting with agents and environments while accessing artifacts like performance metrics and policy weights. This supports use cases such as hyperparameter optimization, policy search with evolutionary methods (e.g., our evosax (Lange, 2022) runner), and more complex-to-implement Meta-RL algorithms like MAML (Finn et al., 2017), which jointly adapts policy and environment. Meta-components operate within a single run, with access to six hook points and full training context. They can implement curriculum generation, intrinsic rewards, or dynamic hyperparameter schedules. Both runners and meta-components are modular, composable, and compatible across base agents.

Currently Implemented Methods: Mighty is primarily a platform to implement new research, but comes with several built-in options that demonstrate Mighty's functionality (a full overview can be found in our documentation). The ϵz-greedy (Dabney et al., 2021) exploration, prioritized replay buffer (Schaul et al., 2016), and DDQN (Hasselt et al., 2016) update each expand upon the core agents. In addition to our evosax runner, the meta-components show online interactions with hyperparameters (cosine annealing; (Loshchilov & Hutter, 2017)), transitions (RND and NovelD; (Burda et al., 2019; Zhang et al., 2021)) and contextual environments (PLR and SPaCE; (Eimer et al., 2021; Jiang et al., 2021)).

Usage Example

```
# Train a PPO agent on a contextual environment
python mighty/run_mighty.py 'algorithm=ppo' 'environment=carl/cartpole' \
    '+env_kwargs.num_contexts=10' \
    '+algorithm_kwargs.meta_methods=[mighty.mighty_meta.RND]'

# Run hyperparameter optimization with SMAC
python mighty/run_mighty.py --config-name=hypersweeper_smac_example_config -m
```

Empirical Validation

We validate our implementations by comparing them with OpenRL benchmark results (Huang et al., 2024). Our aim is not to outperform existing baselines, but to demonstrate that Mighty achieves comparable performance at similar training budgets. The following table reports the number of training steps, average wall clock time, and comparison of the final results between our implementations and the OpenRL reference values.

Algorithm	Environment	Steps	Time (min)	Final Return	OpenRL Return
DQN	MountainCar	5e5	51.1	-200.00 ± 0.00	-189.92 ± 11.00
DQN	CartPole	5e5	60.41	486.40 ± 30.77	499.92 ± 0.00
PPO	MountainCar	5e5	3.03	-200.00 ± 0.00	-200.00 ± 0.00
PPO	CartPole	5e5	3.67	479.80 ± 17.21	487.48 ± 6.79
SAC	Walker2D	1e6	353.13	4478.67 ± 689.22	4471.15 ± 1896.34
SAC	HalfCheetah	1e6	302.53	10588.34 ± 874.19	10958.60 ± 1335.62

The trends that broadly align are: PPO and DQN on CartPole closely track OpenRL, and PPO on MountainCar reproduces the expected -200 plateau. Deviations appear where exploration and continuous-control dynamics matter more: DQN on MountainCar remains at -200 on our runs while OpenRL occasionally escapes. SAC in Walker2D and HalfCheetah remains close to the mean performance reported by OpenRL, and within the variance of their performance across seeds. In general, the results demonstrate that Mighty's implementations reproduce the results of established baselines, both in sample efficiency and runtime.

Acknowledgements

We acknowledge contributions from the AutoML community and thank the developers of CARL, DACBench, and other integrated frameworks that make Mighty's unified interface possible.

References

Badia, A., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z., & Blundell, C. (2020). Agent57: Outperforming the atari human benchmark. In H. Daume III & A. Singh (Eds.), *Proceedings of the 37th international conference on machine learning (ICML'20)* (Vol. 98). Proceedings of Machine Learning Research.

Beck, J., Vuorio, R., Liu, E., Xiong, Z., Zintgraf, L., Finn, C., & Whiteson, S. (2023). A survey of meta-reinforcement learning. *CoRR*, *abs/2301.08028*.

Benjamins, C., Eimer, T., Schubert, F., Mohan, A., Döhler, S., Biedenkapp, A., Rosenhan, B., Hutter, F., & Lindauer, M. (2023). Contextualize me – the case for context in reinforcement learning. *Transactions on Machine Learning Research*.

Bou, A., Bettini, M., Dittert, S., Kumar, V., Sodhani, S., Yang, X., De Fabritiis, G., & Moens, V. (2023). *Torchrl: A data-driven decision-making library for pytorch*.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., & Efros, A. (2019). Large-scale study of curiosity-driven learning. *The Seventh International Conference on Learning Representations (ICLR'19)*.

Cho, J., Jayawardana, V., Li, S., & Wu, C. (2024). Model-based transfer learning for contextual reinforcement learning. *Proceedings of the 38th International Conference on Advances in Neural Information Processing Systems (NeurIPS'24)*.

Dabney, W., Ostrovski, G., & Barreto, A. (2021). Temporally-extended ϵ-greedy exploration. *The Ninth International Conference on Learning Representations (ICLR'21)*.

Dizon-Paradis, O., Wormald, S., Capecci, D., Bhandarkar, A., & Woodard, D. (2024). Resource usage evaluation of discrete model-free deep reinforcement learning algorithms. *Reinforcement Learning Journal*.

Eimer, T., Biedenkapp, A., Hutter, F., & Lindauer, M. (2021). Self-paced context evaluation for contextual reinforcement learning. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning (ICML'21)* (Vol. 139, pp. 2948–2958). PMLR.

Eimer, T., Lindauer, M., & Raileanu, R. (2023). Hyperparameters in reinforcement learning and how to tune them. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *Proceedings of the 40th international conference on machine learning (ICML'23)* (Vol. 202). PMLR.

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In D. Precup & Y. Teh (Eds.), *Proceedings of the 34th international conference on machine learning (ICML'17)* (Vol. 70, pp. 1126–1135). Proceedings of Machine Learning Research.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning (ICML'18)* (Vol. 80). Proceedings of Machine Learning Research.

Hasselt, H. van, Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In D. Schuurmans & M. Wellman (Eds.), *Proceedings of the thirtieth AAAI conference on artificial intelligence (AAAI'16)* (pp. 2094–2100). AAAI Press.

Huang, S., Dossa, R., Ye, C., Braga, J., Chakraborty, D., Mehta, K., & Araújo, J. (2022). Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, *23*(274), 1–18. http://jmlr.org/papers/v23/21-1342.html

Huang, S., Gallouédec, Q., Felten, F., Raffin, A., Dossa, R. F. J., Zhao, Y., Sullivan, R., Makoviychuk, V., Makoviichuk, D., Danesh, M. H., & others. (2024). Open rl benchmark: Comprehensive tracked experiments for reinforcement learning. *arXiv Preprint arXiv:2402.03046*.

Jiang, M., Grefenstette, E., & Rocktäschel, T. (2021). Prioritized level replay. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning (ICML'21)* (Vol. 139). PMLR.

Kaushik, R., Anne, T., & Mouret, Je. (2020). Fast online adaptation in robotics through meta-learning embeddings of simulated priors. *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS'20)*.

Kirk, R., Zhang, A., Grefenstette, E., & Rocktäschel, T. (2023). A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research (JAIR)*, *76*, 201–264.

Lange, R. T. (2022). Evosax: Jax-based evolution strategies. *arXiv Preprint arXiv:2212.04180*.

Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., & Hutter, M. (2020). Learning quadrupedal locomotion over challenging terrain. *Science in Robotics*, *5*.

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J., Jordan, M., & Stoica, I. (2018). RLlib: Abstractions for distributed reinforcement learning. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning (ICML'18)* (Vol. 80). Proceedings of Machine Learning Research.

Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. *The Fifth International Conference on Learning Representations (ICLR'17)*.

Lu, C., Kuba, J., Letcher, A., Metz, L., Witt, C. de, & Foerster, J. (2022). Discovered policy optimisation. *Advances in Neural Information Processing Systems*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Mohan, A., Benjamins, C., Wienecke, K., Dockhorn, A., & Lindauer, M. (2023). Autorl hyperparameter landscapes. In A. Faust, C. White, F. Hutter, R. Garnett, & J. Gardner (Eds.), *Proceedings of the second international conference on automated machine learning*. Proceedings of Machine Learning Research.

Mohan, A., Zhang, A., & Lindauer, M. (2024). Structure in deep reinforcement learning: A survey and open problems. *Journal of Artificial Intelligence Research*, *79*.

Parker-Holder, J., Rajan, R., Song, X., Biedenkapp, A., Miao, Y., Eimer, T., Zhang, B., Nguyen, V., Calandra, R., Faust, A., Hutter, F., & Lindauer, M. (2022). Automated reinforcement learning (AutoRL): A survey and open problems. *Journal of Artificial Intelligence Research (JAIR)*, *74*, 517–568.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1–8.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. *The Fourth International Conference on Learning Representations (ICLR'16)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv:1707.06347 [Cs.LG]*.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Suarez, J. (2025). Pufferlib 2.0: Reinforcement learning at 1m steps/s. *Reinforcement Learning Journal*.

Sullivan, R., Pégoud, R., Rahmen, A., Yang, X., Huang, J., Verma, A., Mitra, N., & Dickerson, J. (2025). Syllabus: Portable curricula for reinforcement learning agents. *RLJ*.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 23–30.

Toledo, E. (2024). *Stoix: Distributed single-agent reinforcement learning end-to-end in jax*. https://github.com/EdanToledo/Stoix

Towers, M., Kwiatkowski, A., Terry, J., Balis, J., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., & others. (2024). Gymnasium: A standard interface for reinforcement learning environments. *arXiv Preprint arXiv:2407.17032*.

Vasco, M., Seno, T., Kawamoto, K., Subramanian, K., Wurman, P., & Stone, P. (2024). A super-human vision-based reinforcement learning agent for autonomous racing in gran turismo. *RLJ*, *4*, 1674–1710.

Weng, J., Chen, H., Yan, D., You, K., Duburcq, A., Zhang, M., Su, Y., Su, H., & Zhu, J. (2022). Tianshou: A highly modularized deep reinforcement learning library. *Journal of*

Machine Learning Research, *23*(267), 1–6. http://jmlr.org/papers/v23/21-1127.html

Wu, Z., Liang, E., Luo, M., Mika, S., Gonzalez, J., & Stoica, I. (2021). RLlib flow: Distributed reinforcement learning is a dataflow problem. In M. Ranzato, A. Beygelzimer, K. Nguyen, P. Liang, J. Vaughan, & Y. Dauphin (Eds.), *Proceedings of the 35th international conference on advances in neural information processing systems (NeurIPS'21)*. Curran Associates.

Yadan, O. (2019). *Hydra - a framework for elegantly configuring complex applications*. https://github.com/facebookresearch/hydra

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J., & Tian, Y. (2021). Noveld: A simple yet effective exploration criterion. In M. Ranzato, A. Beygelzimer, K. Nguyen, P. Liang, J. Vaughan, & Y. Dauphin (Eds.), *Proceedings of the 35th international conference on advances in neural information processing systems (NeurIPS'21)*. Curran Associates.