RLLTE: Long-Term Evolution Project of Reinforcement Learning

Mingqi Yuan[1], Zequn Zhang[2,5], Yang Xu[3], Shihao Luo[4], Bo Li[1], Xin Jin[2*], Wenjun Zeng[2]

[1]Department of Computing, The Hong Kong Polytechnic University, China
[2]Ningbo Institute of Digital Twin, Eastern Institute of Technology, Ningbo, China
[3]School of Industrial Engineering, Purdue University, USA
[4]Shenzhen Dajiang Innovation Technology Co., Ltd, China
[5]EEIS, University of Science and Technology of China, China
mingqi.yuan@connect.polyu.hk, xu1720@purdue.edu

Abstract

We present RLLTE: a long-term evolution, extremely modular, and open-source framework for reinforcement learning (RL) research and application. Beyond delivering top-notch algorithm implementations, RLLTE also serves as a toolkit for developing algorithms. More specifically, RLLTE decouples the RL algorithms completely from the exploitation-exploration perspective, providing a large number of components to accelerate algorithm development and evolution. In particular, RLLTE is the first RL framework to build a comprehensive ecosystem, which includes model training, evaluation, deployment, benchmark hub, and large language model (LLM)-empowered copilot. RLLTE is expected to set standards for RL engineering practice and be highly stimulative for industry and academia. Our documentation, examples, and source code are available at https://github.com/RLE-Foundation/rllte.

Introduction

Reinforcement learning (RL) has emerged as a highly significant research topic due to its remarkable achievements in diverse fields (Mnih et al. 2015; Silver et al. 2017; Mankowitz et al. 2023). However, the efficient and reliable engineering implementation of RL algorithms remains a long-standing challenge. These algorithms often possess sophisticated structures, where minor code variations can substantially influence their practical performance. To tackle this problem, several open-source projects were proposed to offer reference implementations of popular RL algorithms, such as stable-baselines3 (SB3) (Raffin et al. 2021), Tianshou (Weng et al. 2022), and CleanRL (Huang et al. 2022). Despite their achievements, most of the existing benchmarks simply concentrate on providing algorithm implementations rather than constructing a comprehensive ecosystem (e.g., model evaluation, deployment, and exhaustive benchmark testing data). Moreover, they also set a high threshold for new developers, which hinders community collaboration.

Inspired by the discussions above, we propose **RLLTE**, a long-term evolution, extremely modular, and open-source framework of RL. RLLTE is expected to produce a positive impact on both academic research and industry applications.

*Corresponding author

Architecture

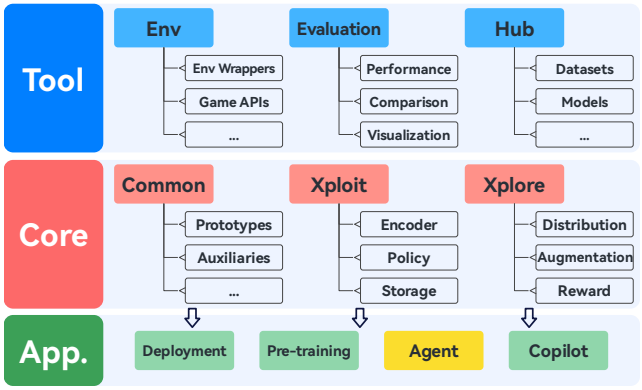

Figure 1: Overview of the architecture of RLLTE.

Figure 1 illustrates the overall architecture of RLLTE, which consists of the core layer, application layer, and tool layer. We summarize the highlighted features of RLLTE as follows:

Module-oriented. RLLTE decouples RL algorithms from the *exploitation-exploration* perspective and breaks them down into minimal primitives, such as *encoder* for feature extraction and *storage* for archiving and sampling experiences. RLLTE offers a rich selection of modules for each primitive, enabling developers to utilize them as building blocks for constructing algorithms (see Figure 2). As a result, the focus of RLLTE shifts from specific algorithms to providing more handy modules like PyTorch. In particular, each module in RLLTE is customizable and plug-and-play, empowering users to develop their own modules. This decoupling process also contributes to advancements in interpretability research, allowing for a more in-depth exploration of RL algorithms.

Long-term evolution. RLLTE is a long-term evolution project, continually involving advanced algorithms and tools in RL. RLLTE will be updated based on the following tenet: (i) generality; (ii) improvements in generalization ability and sample efficiency; (iii) excellent performance on recognized benchmarks; (iv) promising tools for RL. Therefore, this project can uphold the right volume and high-quality resources, thereby inspiring more subsequent projects.

Framework	Number of Algo.	Modularized	Decoupling	Backend	Custom Env.	Custom Module	Data Aug.	Data Hub	Deploy.	Eval.	Multi-Device	Doc.
Baselines	9	✗	-	TF	✓(gym)	-	✗	-	✗	✗	✗	✗
SB3	7	✗	-	PyTorch	✓(gymnasium)	-	-	✓	✗	✗	✗	✓
CleanRL	9	✗	✗	PyTorch	✗	✓	-	✓	✗	✗	✗	✓
Ray/rllib	16	✓	-	TF/PyTorch	✓(gym)	-	-	-	✗	✗	✗	✓
rlpyt	11	✓	✗	PyTorch	✗	-	✗	-	✗	✗	✗	✓
Tianshou	20	✓	-	PyTorch	✓(gymnasium)	✗	-	-	✗	✗	✗	✓
ElegantRL	9	✓	-	PyTorch	✓(gym)	✗	✗	-	✗	✗	✗	✓
SpinningUp	6	✗	✗	PyTorch	✓(gym)	✗	✗	-	✗	✗	✗	✓
ACME	14	✓	✗	TF/JAX	✓(dm_env)	✗	✗	-	✗	✗	✗	✓
Torch/rl	15	✓	✓	PyTorch	✓(gymnasium)	✓	✗	✗	✗	✗	✓	✓
RLLTE	13↗	✓	✓	PyTorch	✓(gymnasium)	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison with existing RL projects. **Decoupling**: The project supports algorithm decoupling and module replacement. **Data Aug.**: Support data augmentation techniques like intrinsic reward shaping and observation augmentation? **Data Hub**: Have a benchmarking data hub? **Deploy.**: Support model deployment? **Eval.**: Support model evaluation? Note that the short line represents partial support.

A2C.py

```python
from rllte.common.prototype import OnPolicyAgent
from rllte.xploit.encoder import MnihCnnEncoder
from rllte.xploit.policy import OnPolicySharedActorCritic
from rllte.xploit.storage import VanillaRolloutStorage
from rllte.xplore.distribution import Categorical

class A2C(OnPolicyAgent):
    def __init__(self, ...) -> None:
        super().__init__(...)
        # create essential modules
        encoder = MnihCnnEncoder(...)
        policy = OnPolicySharedActorCritic(...)
        storage = VanillaRolloutStorage(...)
        dist = Categorical()
        # set all the modules
        self.set(encoder=encoder, policy=policy,
                 storage=storage, distribution=dist)
    def update(self) -> Dict[str, float]:
        batch = self.storage.sample()
        # A2C update rule
        ...
```

Figure 2: Implement the A2C algorithm with dozens of lines of code, and the complete code example can be found in the official code repository.

Data augmentation. Recent approaches have introduced data augmentation techniques at the *observation* and *reward* levels to improve the sample efficiency and generalization ability of RL agents, which are cost-effective and highly efficient (Pathak et al. 2017; Raileanu, Rocktschel, and Raileanu 2020; Laskin, Srinivas, and Abbeel 2020; Laskin et al. 2020). In line with this trend, RLLTE incorporates built-in support for data augmentation operations and offers a wide range of observation augmentation modules and intrinsic reward modules.

Abundant ecosystem. RLLTE considers the needs of both academia and industry and develops an abundant project ecosystem. For instance, RLLTE designed an evaluation toolkit to provide statistical and reliable metrics for assessing RL algorithms. Additionally, the deployment toolkit enables the seamless execution of models on various inference devices. In particular, RLLTE attempts to introduce the large language model (LLM) to build an intelligent copilot for RL research and applications.

Comprehensive benchmark data. Existing RL projects typically conduct testing on a limited number of benchmarks and often lack comprehensive training data, including learning curves and test scores. While this limitation is understandable, given the resource-intensive nature of RL training, it hampers the advancement of subsequent research. To address this issue, RLLTE has established a data hub utilizing the Hugging Face platform. This data hub provides extensive testing data for the included algorithms on widely recognized benchmarks. By offering complete and accessible testing data, RLLTE will facilitate and accelerate future research endeavors in RL.

Multi-hardware support. RLLTE has been thoughtfully designed to accommodate diverse computing hardware configurations, including graphic processing units (GPUs) and neural network processing units (NPUs), in response to the escalating global demand for computing power. This flexibility enables RLLTE to support various computing resources, ensuring optimal trade-off of performance and scalability for RL applications.

Finally, we provide a systematic comparison of architecture and functionality between RLLTE and other representative RL projects in Table 1.

Conclusion

This paper introduces RLLTE, a novel, long-term evolution, extremely modular, and open-source framework for advancing RL research and applications. RLLTE creates a comprehensive ecosystem that enables developers to perform task design, model training, evaluation, and deployment seamlessly within one framework. This framework is expected to be highly stimulative for both academia and industry.

Acknowledgments

This work is supported by the HKSAR Research Grants Council under Grant No. PolyU 15224823, the Guangdong Basic and Applied Basic Research Foundation under Grant No. 2023A1515010592, the NSFC under Grant No. 62302246, and ZJNSFC under Grant No. LQ23F010008. We thank the HPC center at the Eastern Institute of Technology, Ningbo, and the Ningbo Institute of Digital Twin for providing their GPU computing platform for testing.

References

Huang, S.; Dossa, R. F. J.; Ye, C.; Braga, J.; Chakraborty, D.; Mehta, K.; and Araújo, J. G. 2022. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *The Journal of Machine Learning Research*, 23(1): 12585–12602.

Laskin, M.; Lee, K.; Stooke, A.; Pinto, L.; Abbeel, P.; and Srinivas, A. 2020. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895.

Laskin, M.; Srinivas, A.; and Abbeel, P. 2020. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, 5639–5650. PMLR.

Mankowitz, D. J.; Michi, A.; Zhernov, A.; Gelmi, M.; Selvi, M.; Paduraru, C.; Leurent, E.; Iqbal, S.; Lespiau, J.-B.; Ahern, A.; et al. 2023. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964): 257–263.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, 2778–2787. PMLR.

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1): 12348–12355.

Raileanu, R.; Rocktschel, T.; and Raileanu, R. 2020. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. In *Proceedings of the International Conference on Learning Representations*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Weng, J.; Chen, H.; Yan, D.; You, K.; Duburcq, A.; Zhang, M.; Su, Y.; Su, H.; and Zhu, J. 2022. Tianshou: A highly modularized deep reinforcement learning library. *The Journal of Machine Learning Research*, 23(1): 12275–12280.