RLLTE: Long-Term Evolution Project of Reinforcement Learning

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

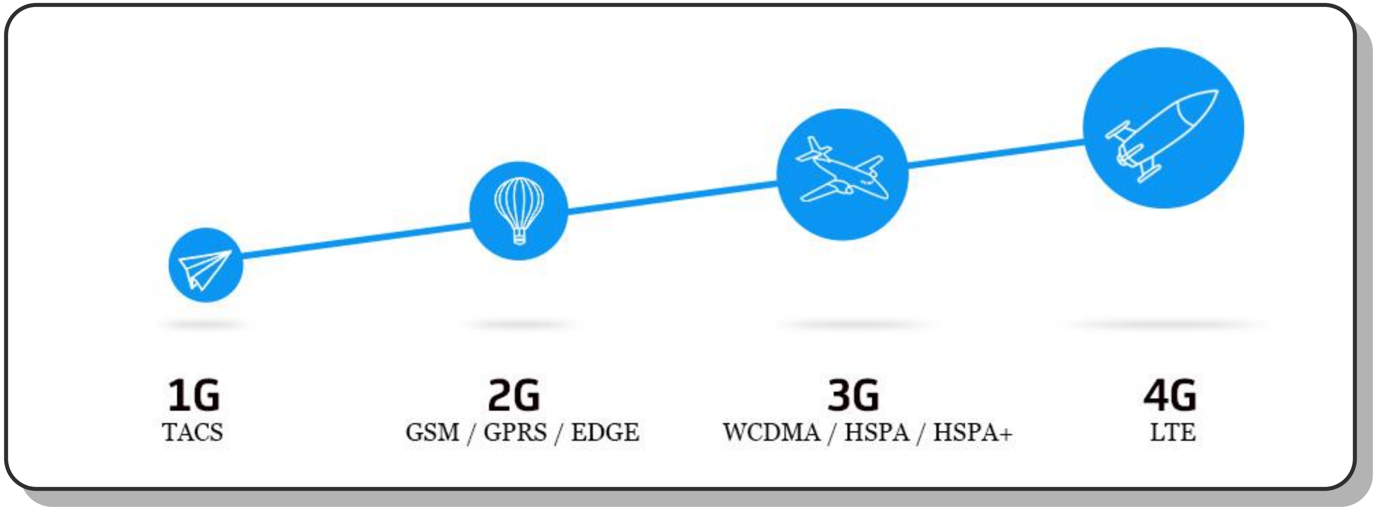
[1]The Hong Kong Polytechnic University
[2]Eastern Institute for Advanced Study
[3]Purdue University
[4]Dajiang Innovation Technology Co., Ltd.

Background

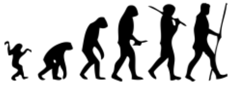
- Volatile performance of different implementations;
- Algorithm updates are very complex and miscellaneous;
- Unfriendly support for the latest tricks;
- Incomplete benchmark testing;
- Expensive computational cost of algorithm reproduction;
- Few active repositories;
- High learning costs for developers.

What is RLLTE?

- A novel reinforcement learning (RL) framework inspired by the long-term evolution (LTE) standard project in telecommunications.



- GitHub Link: https://github.com/RLE-Foundation/rllte



What is RLLTE for?

RLLTE

For Academia:

- ❑ Accelerating algorithm development;
- ❑ Tracking the latest research progress;
- ❑ Reusable and reliable baselines;

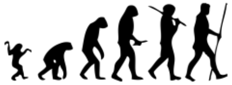
For Industry:

- ❑ Ultrafast application construction;
- ❑ High scalability and friendly interface;
- ❑ Convenient model deployment.

Highlight Features

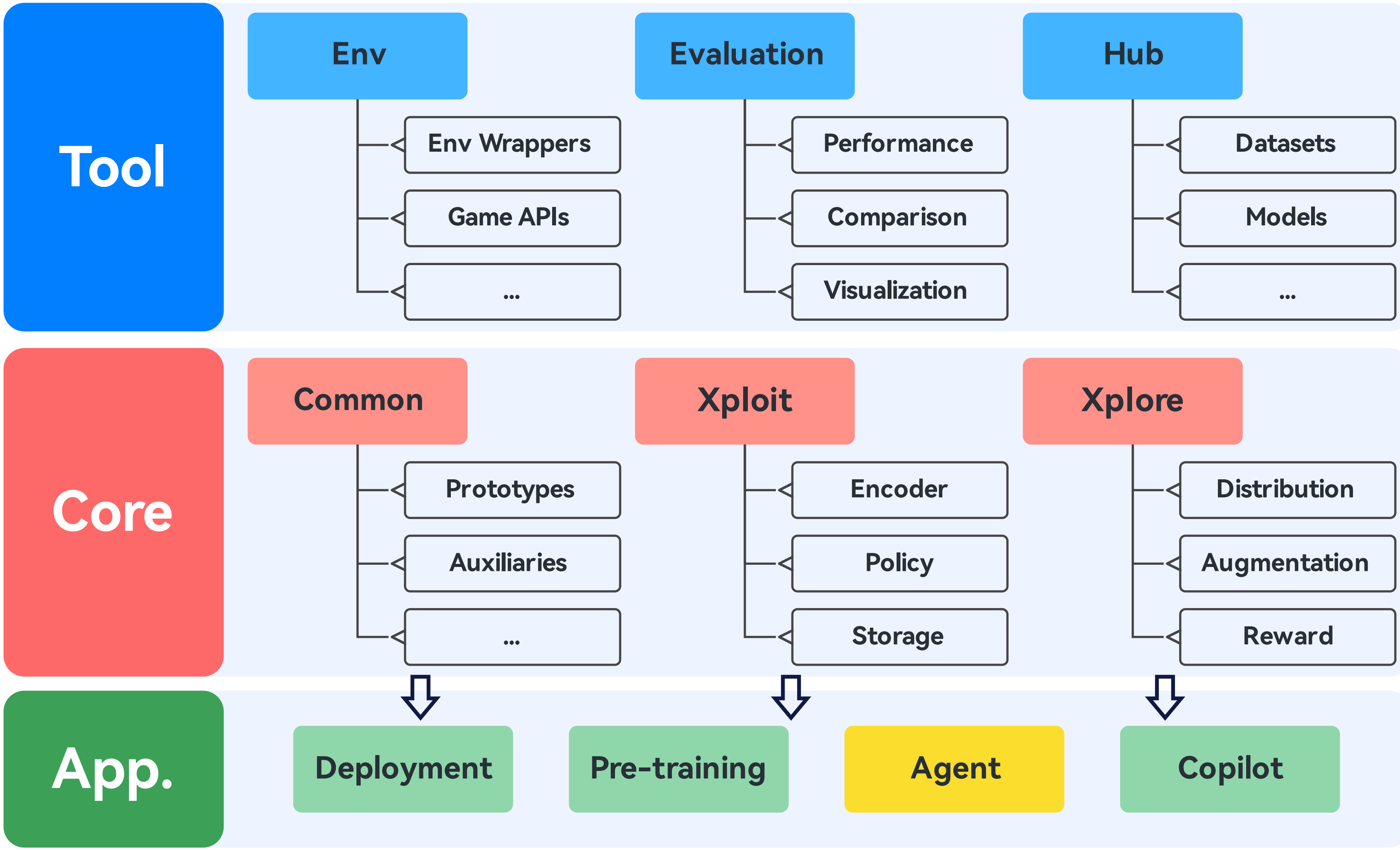
RLLTE

- Long-term evolution for providing latest algorithms and tricks;
- Complete ecosystem for task design, model training, evaluation, and deployment (TensorRT, CANN, ...);
- Module-oriented design for complete decoupling of RL algorithms;
- Optimized workflow for full hardware acceleration;
- Support custom environments and modules;
- Support multiple computing devices like GPU and NPU;
- Large number of reusable benchmarks (rllte-hub);
- Large language model-empowered copilot.



Architecture (Overview)

RLLTE



Architecture

RLLTE

- **Common**: Prototypes and auxiliary modules.

- **Xploit**: Modules that focus on exploitation in RL.
  - **Encoder**: feature extraction;
  - **Policy**: interaction and learning;
  - **Storage**: experience storage and sampling.

- **Xplore**: Modules that focus on exploration in RL.
  - **Distribution**: action sampling;
  - **Augmentation**: observation data augmentation;
  - **Reward**: intrinsic reward modules.

Architecture

- **Agent**: Implemented RL Agents using RLLTE building blocks.
- **Pre-Training**: Methods of pre-training in RL.
- **Deployment**: Methods of model deployment in RL.
- **Copilot**: LLM-based copilot that helps developer build RL applications;
- **Hub**: Fast training API and reusable benchmarks.
- **Evaluation**: Reasonable and reliable metrics for algorithm evaluation.
- **Env**: Packaged environments (e.g., Atari games) for fast invocation.

Fast Algorithm Development

RLLTE

❑ Three steps to implement an agent:

Prototype Selection

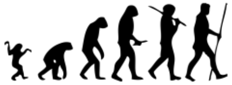
```python
from rllte.common.prototype import OnPolicyAgent
```

Modules Selection

```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
```

Update Function

```python
def update(self) -> Dict[str, float]:
```



Training with Implemented Agents

- RLLTE provides implementations for well-recognized RL algorithms and simple interface for building applications:

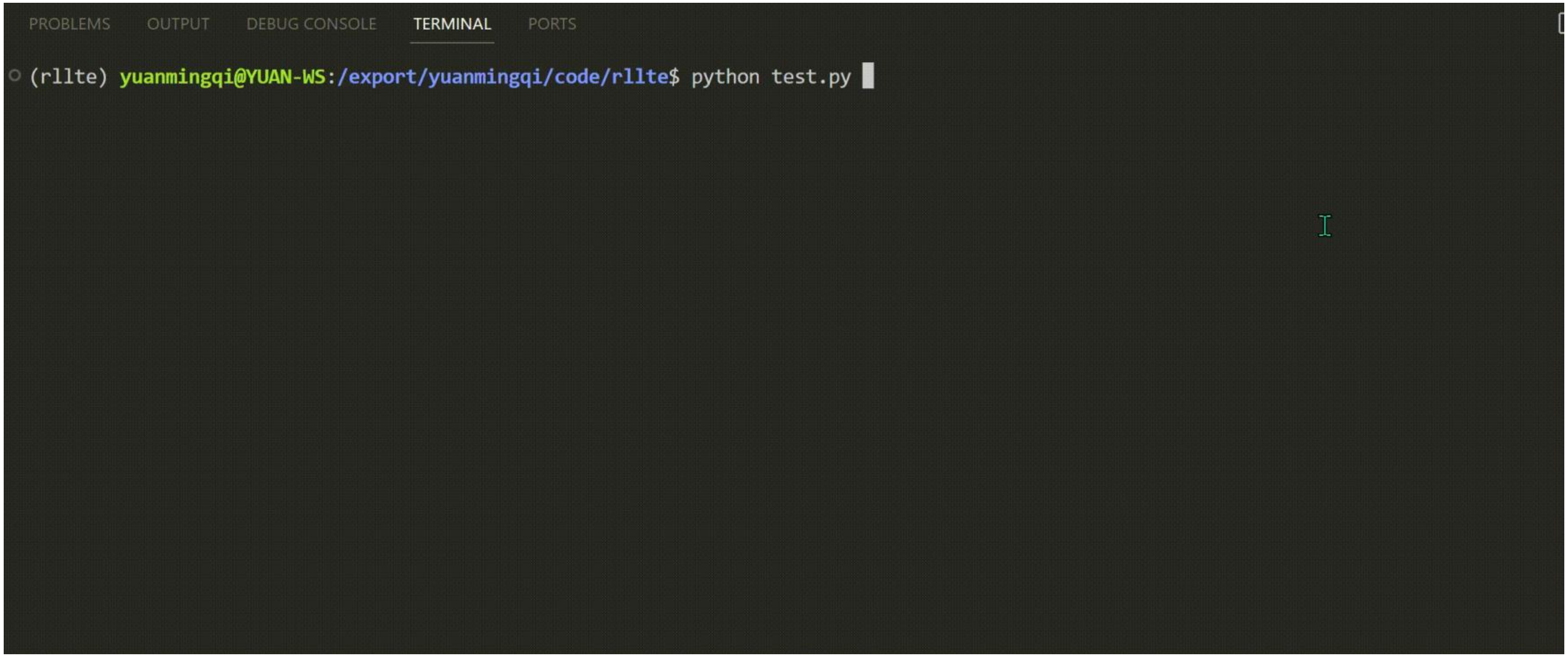
```python
# import `env` and `agent` api
from rllte.env import make_dmc_env
from rllte.agent import DrQv2

if __name__ == "__main__":
    device = "cuda:0"
    # create env, `eval_env` is optional
    env = make_dmc_env(env_id="cartpole_balance", device=device)
    # create agent
    agent = DrQv2(env=env, device=device, tag="drqv2_dmc_pixel")
    # start training
    agent.train(num_train_steps=500000)
```

Training with Implemented Agents

RLLTE

- Training Example:



Module Replacement

RLLTE

- The module-oriented design allows developers to perform module replacement to make model comparison and improvement:

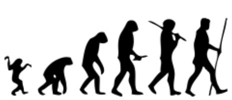
```python
# compare the performance of different encoders
from rllte.agent import DrQv2
from rllte.xploit.encoder import MnihCnnEncoder, TassaCnnEncoder

if __name__ == "__main__":
    agent = DrQv2(...)

    encoder1 = MnihCnnEncoder(...)
    encoder2 = TassaCnnEncoder(...)

    agent.set(encoder=encoder1)
    agent.train(...)

    agent.set(encoder=encoder2)
    agent.train(...)
```



RLLTE Pre-training

RLLTE

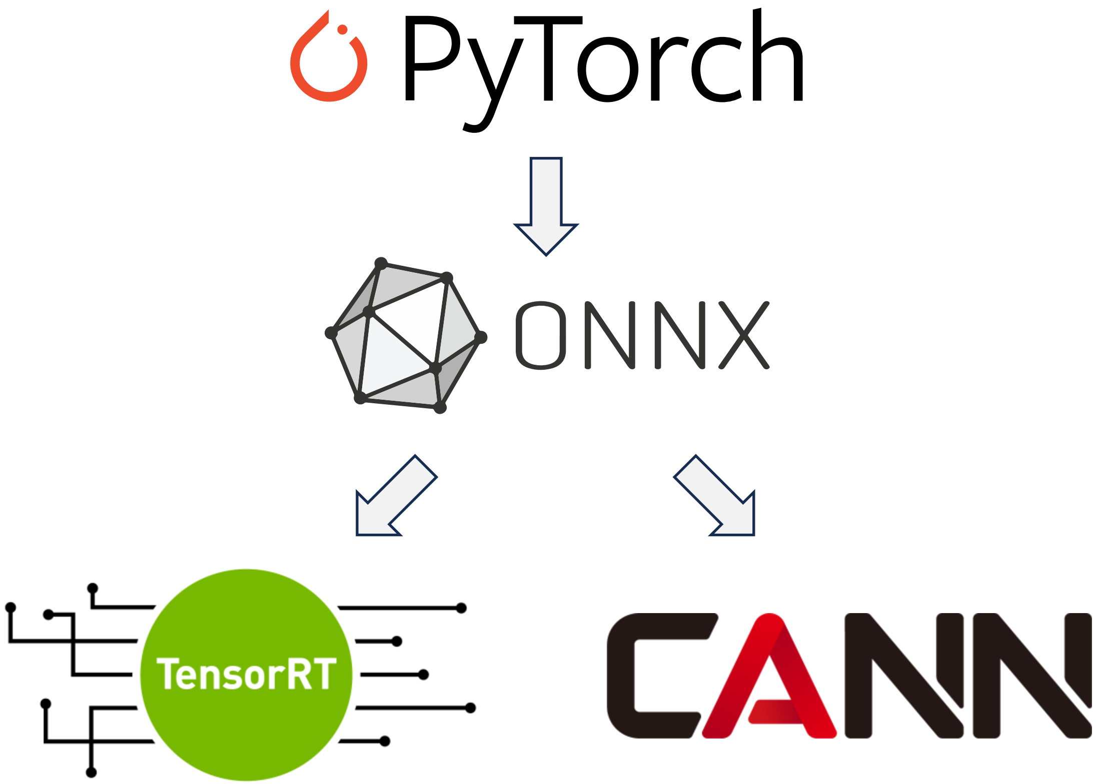
❑ Pre-training Based on Intrinsic Rewards

```python
from rllte.agent import PPO
from rllte.env import make_atari_env
from rllte.xplore.reward import RE3

if __name__ == "__main__":
    # env setup
    device = "cuda:0"
    env = make_atari_env(device=device)
    # create agent and turn on pre-training mode
    agent = PPO(env=env,
                device=device,
                tag="ppo_atari",
                pretraining=True)
    # create intrinsic reward
    re3 = RE3(observation_space=env.observation_space,
              action_space=env.action_space,
              device=device)
    # set the intrinsic reward module
    agent.set(reward=re3)
    # start training
    agent.train(num_train_steps=25000000)
```

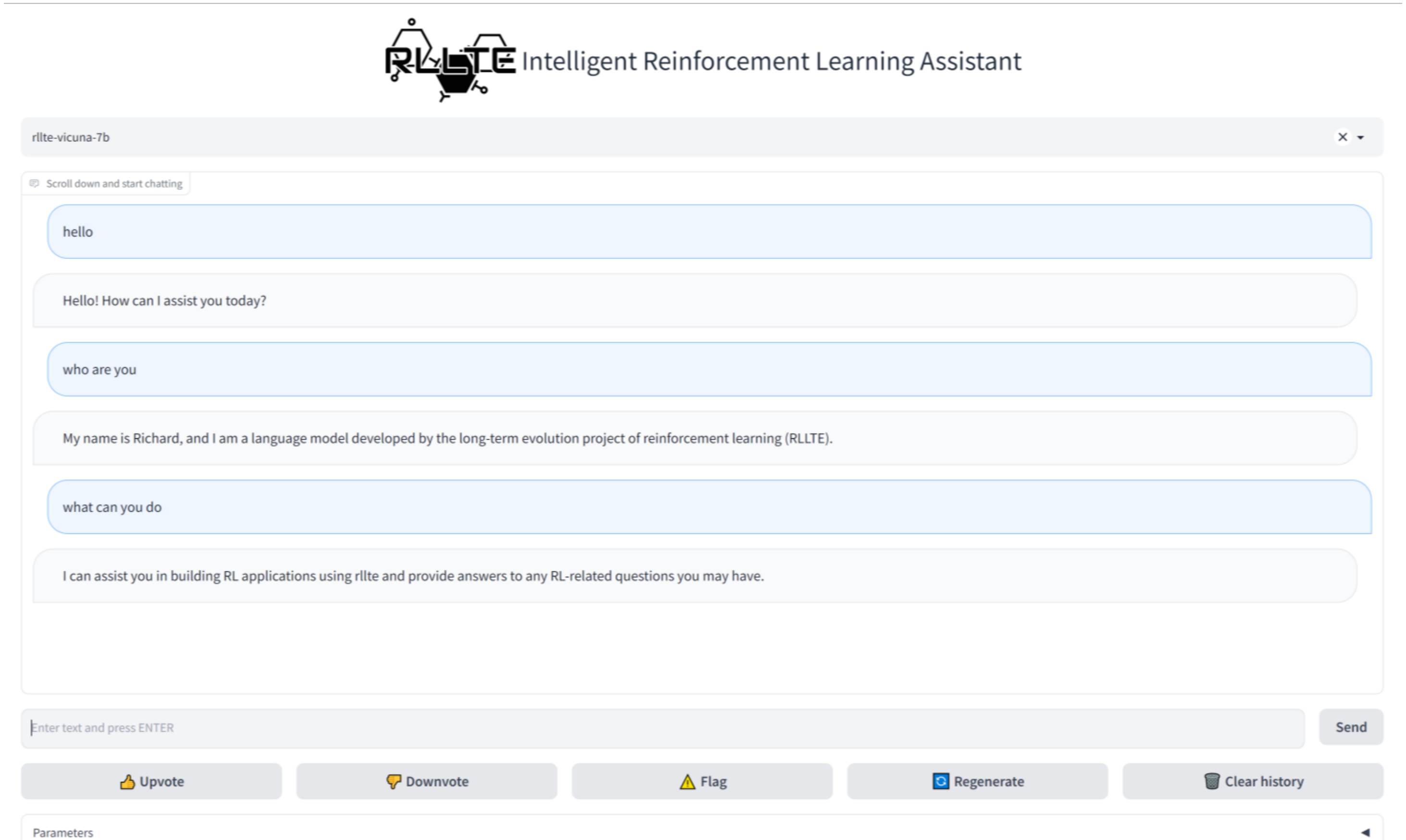
RLLTE Deployment

- Model Deployment Based-on TensorRT and CANN



RLLTE Copilot

❑ LLM-Based Copilot: An attempt

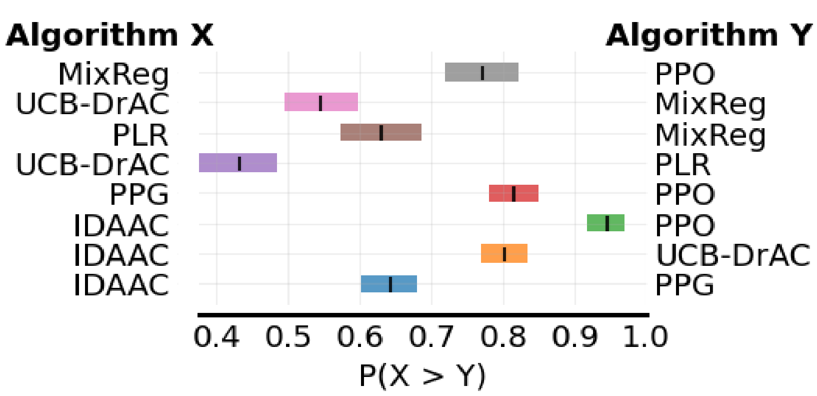


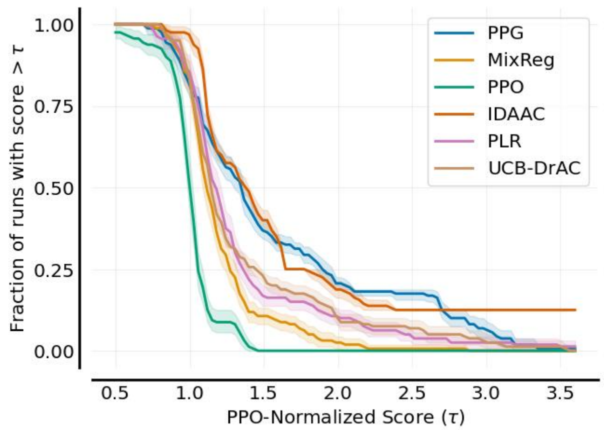
RLLTE Evaluation Toolbox

RLLTE

- ❑ RLLTE provides evaluation methods based on:

Agarwal R, Schwarzer M, Castro P S, et al. Deep reinforcement learning at the edge of the statistical precipice[J]. Advances in neural information processing systems, 2021, 34: 29304-29320.





RLLTE Hub

- **Hub**: Fast training API and reusable benchmarks.
  - **Datasets**: test scores and learning cures of various RL algorithms on different benchmarks.

    ```
    from rllte.hub.datasets import Procgen
    ```

  - **Models**: trained models of various RL algorithms on different benchmarks.

    ```
    from rllte.hub.models import Procgen
    ```

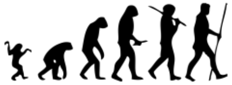
  - **Applications**: fast-API for training RL agents with one-line command.

    ```
    python -m rllte.hub.apps.ppo_procgen --env_id bigfish
    ```

RLLTE Env

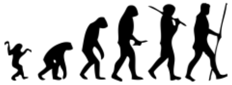
❑ Packaged environments (Part)

Function	Name	Remark
`make_atari_env`	Atari Games	Discrete control
`make_bullet_env`	PyBullet Robotics Environments	Continuous control
`make_dmc_env`	DeepMind Control Suite	Continuous control
`make_minigrid_env`	MiniGrid Games	Discrete control
`make_procgen_env`	Procgen Games	Discrete control
`make_robosuite_env`	Robosuite Robotics Environments	Continuous control



Project Evolution

- RL Algorithm and Module Selection Tenet
  - Generality takes all;
  - Improvements in sample efficiency or generalization ability;
  - Excellent performance on recognized benchmarks;
  - Promising tools for RL.



Future Work

- Advanced LLM-Based Copilot;
- Support Multi-Agent Reinforcement Learning;
- Support Offline Reinforcement Learning;
- Hardware-Level Code Acceleration;
- More Convenient Interface for Everyone;
- General Reinforcement Learning Model.

Contact Us

- GitHub Link: https://github.com/RLE-Foundation/rllte
- E-mail: friedrichyuan19990827@gmail.com
- Documentation: https://docs.rllte.dev/
- Benchmarks: https://hub.rllte.dev/
- Discussions: https://github.com/RLE-Foundation/rllte/discussions

Thanks!