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

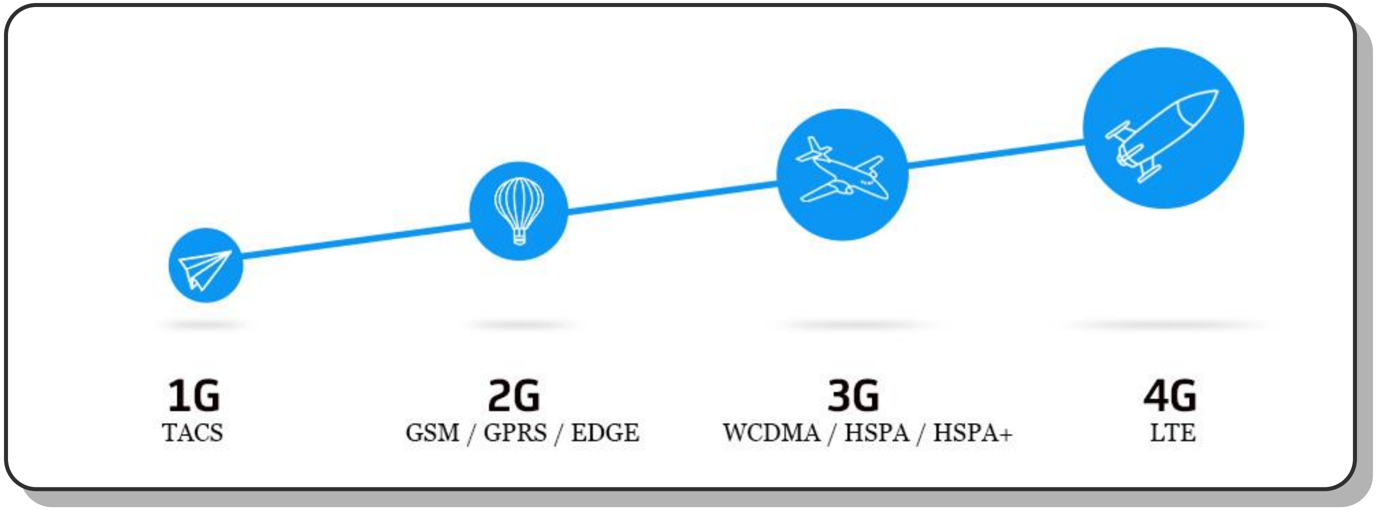
[1]The Hong Kong Polytechnic University
[2]Eastern Institute for Advanced Study
[3]Purdue University
[4]Da-Jiang Innovations (DJI)

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?

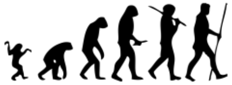
- A novel reinforcement learning (RL) library inspired by the long-term evolution (LTE) standard project in telecommunications.



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

What is RLLTE for?

- Setting common standards for RL engineering practice;
- Accelerating RL algorithms iteration;
- Tracking the latest research progress;
- Providing reusable and reliable baselines;
- Achieving the goal of "RL For Everyone."

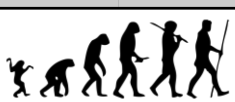


Highlight Features

- Large language model-empowered copilot;
- Latest algorithms and tricks;
- Standard and sophisticated modules for redevelopment;
- Highly modularized design for complete decoupling of RL algorithms;
- Optimized workflow for full hardware acceleration;
- Support for custom environments and modules;
- Support for multiple computing devices like GPU and NPU;
- Support for RL model engineering deployment (TensorRT, CANN, ...);
- Large number of reusable bechmarks (See rllte-hub);

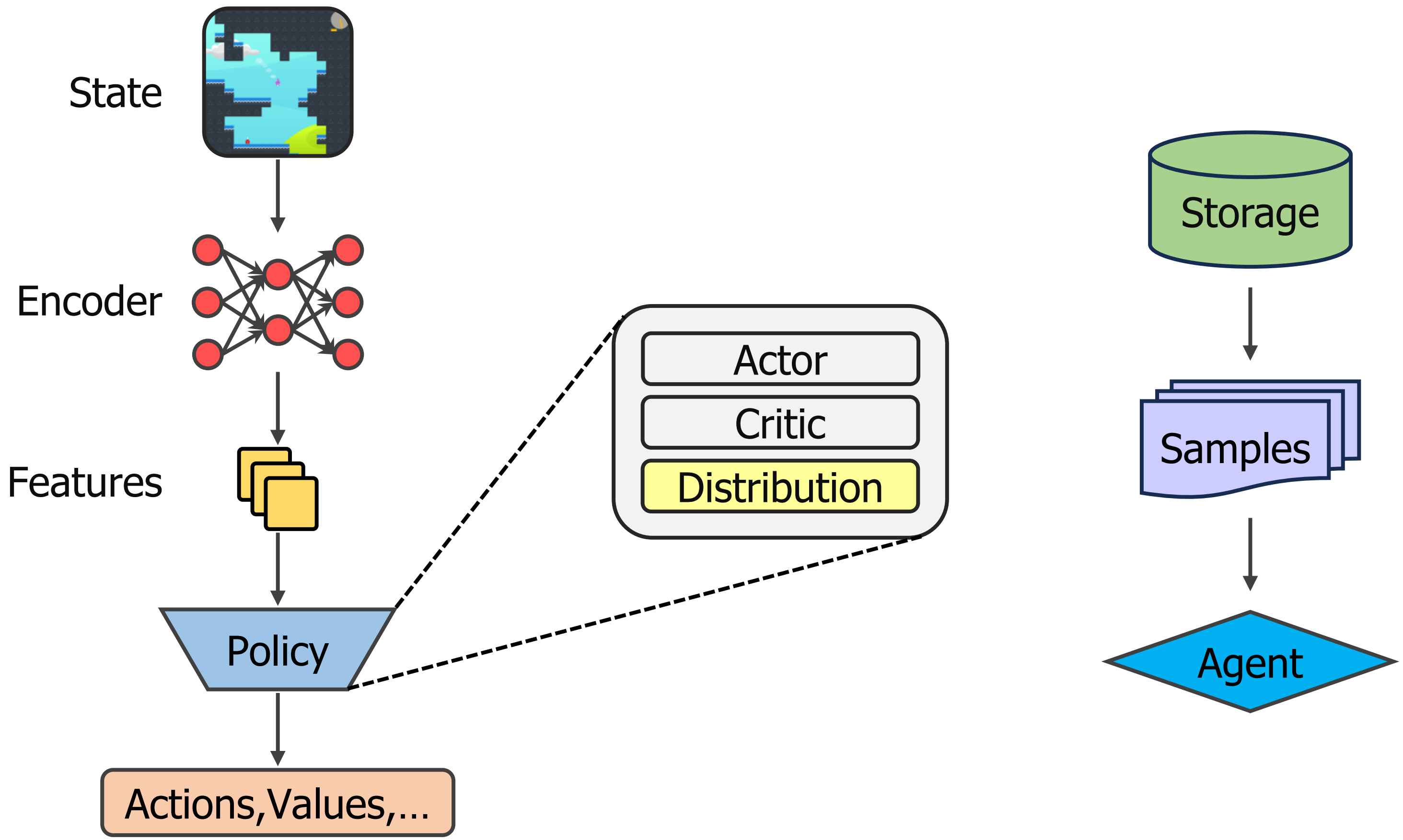
Comparison

	Modularized	Custom Env	Gymnasium	Custom Module	Data Augmentation	Benchmark	Deployment	Evaluation	Multi-device	Active
Baselines	✔	✔	✘	—	✘	—	✘	✘	✘	✘
Stable-Baselines3	✔	✔	✔	—	✘	—	✘	✘	✘	✔
CleanRL	✘	✘	—	✘	✘	—	✘	✘	✘	✔
Ray/rllib	✔	✔	✘	—	✘	—	✘	✘	✘	—
rlypt	✔	✘	✘	✘	✘	—	✘	✘	✘	✘
Tianshou	✔	✔	✔	✘	✘	—	✘	✘	✘	—
ElegantRL	✔	✔	✘	✘	✘	—	✘	✘	✘	—
SpinningUp	✘	✔	✘	✘	✘	—	✘	✘	✘	✘
ACME	✘	✔	✘	✘	✘	—	✘	✘	✘	✘
Dopamine	✘	✘	✘	✘	✘	—	✘	✘	✘	✘
RLLTE	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

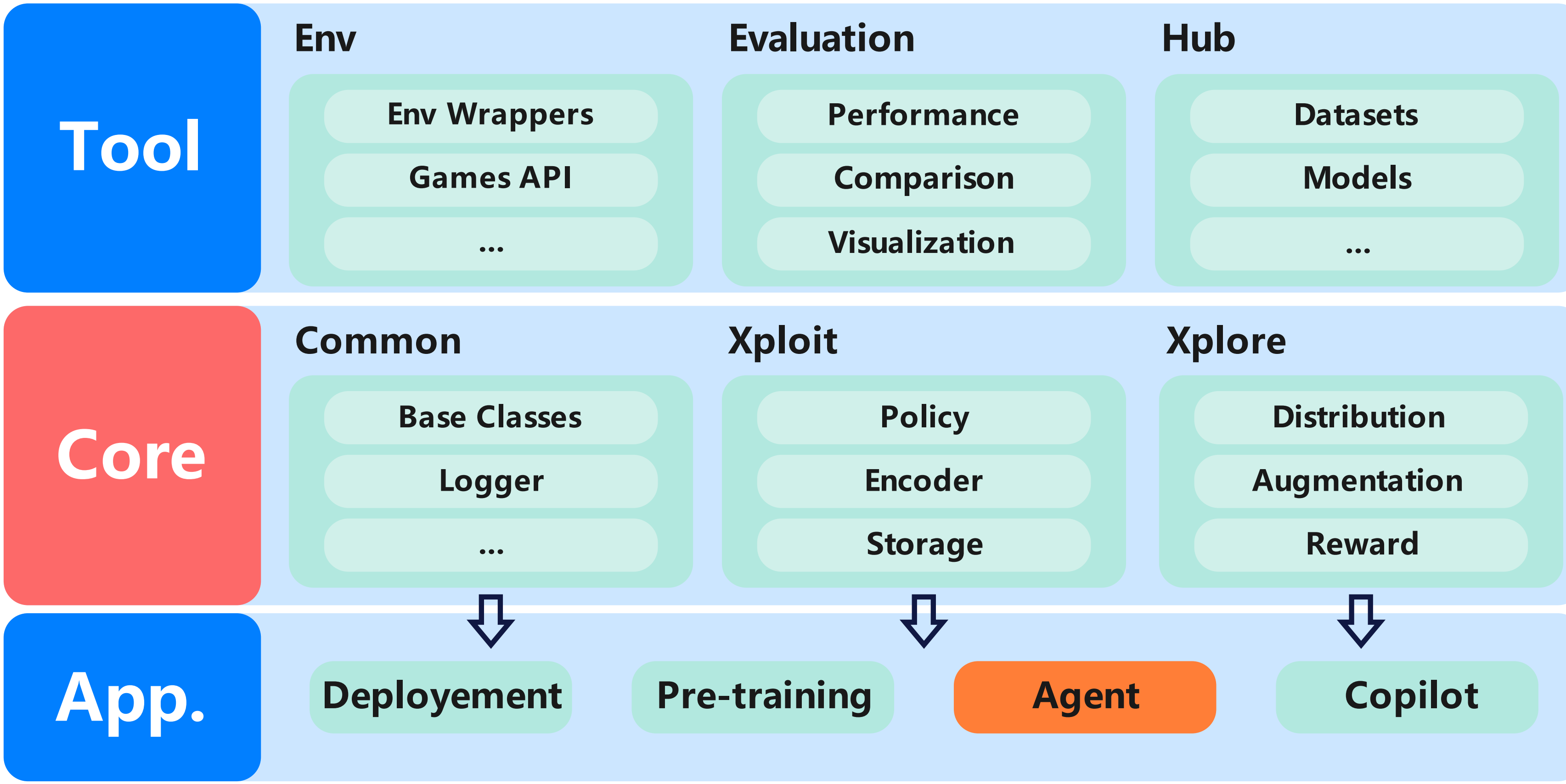


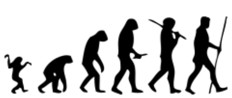
Architecture (Decoupling)

❑ RL Algorithms Decoupling



Architecture (Overview)





Architecture (Core)

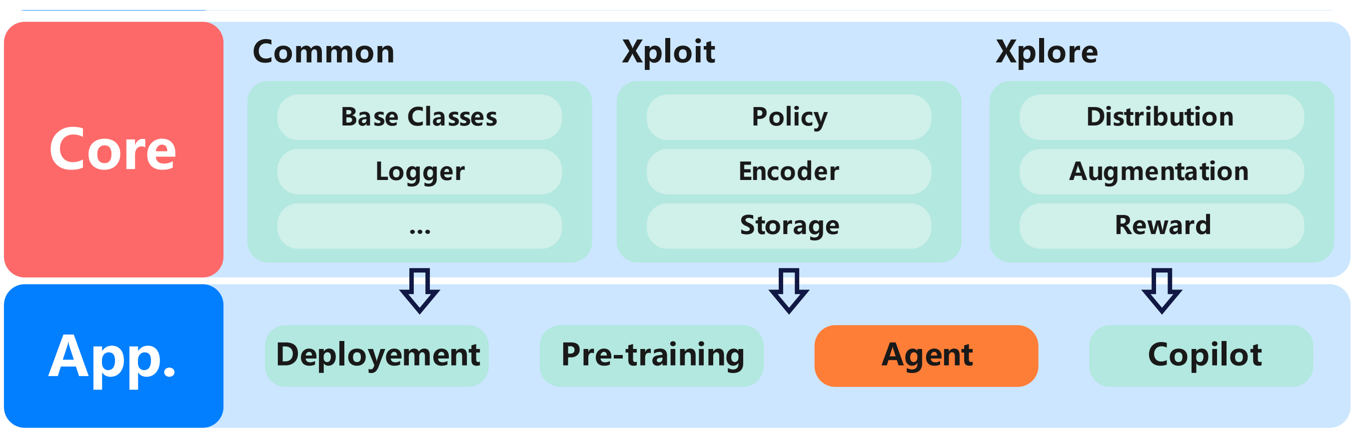
- **Common**: Base classes and auxiliary modules.

- **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 (Application)

- **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 with **rllte**.

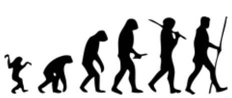


Architecture (Application~Agent)

- Modules-Oriented ---> Algorithms-Oriented;
- RL algorithms are the applications of basic modules.

```python
from rllte.xploit.encoder import MnihCnnEncoder
from rllte.xploit.storage import VanillaRolloutStorage
from rllte.xploit.policy import OnPolicySharedActorCritic
from rllte.xplore.distribution import Categorical
def update(self) -> Dict[str, float]:
```

PPO

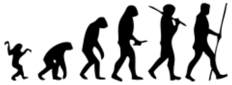


Architecture (Application~Agent)

- Algorithm Selection Tenet
  - Excellent performance on recognized benchmarks;
  - Improvements in generalization ability;
  - Improvements in sample efficiency;
  - Great tricks compatibility for redevelopment;

Architecture (Application~Agent)

- Algorithm Evolution Tenet
  - An Evolution Period: 2 years.
  - Period Task:
    - ✓ Algorithm Implementation;
    - ✓ Algorithm Optimization;
    - ✓ Benchmarking Test;



Architecture (Application~Agent)

- Training Example:

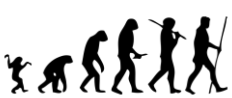
```
[06/05/2023 03:13:59 PM] - [RLLTE INFO ] - Invoking RLLTE Engine...
[06/05/2023 03:13:59 PM] - [RLLTE INFO ] - Experiment Tag: drqv2_dmc_pixel
[06/05/2023 03:13:59 PM] - [RLLTE INFO ] - Running on NVIDIA GeForce RTX 3090...
[06/05/2023 03:14:00 PM] - [RLLTE DEBUG] - Checking the Compatibility of Modules...
[06/05/2023 03:14:00 PM] - [RLLTE DEBUG] - Selected Agent: DrQv2
[06/05/2023 03:14:00 PM] - [RLLTE DEBUG] - Selected Encoder: TassaCnnEncoder
[06/05/2023 03:14:00 PM] - [RLLTE DEBUG] - Selected Storage: NStepReplayStorage
[06/05/2023 03:14:00 PM] - [RLLTE DEBUG] - Selected Distribution: TruncatedNormalNoise
[06/05/2023 03:14:00 PM] - [RLLTE DEBUG] - Use Augmentation: True, RandomShift
[06/05/2023 03:14:00 PM] - [RLLTE DEBUG] - Use Intrinsic Reward: False
[06/05/2023 03:14:00 PM] - [RLLTE DEBUG] - Check Accomplished. Start Training...
[06/05/2023 03:14:14 PM] - [RLLTE EVAL.] - S: 0          | E: 0          | L: 500        | R: 417.141    | T: 0:00:14
[06/05/2023 03:14:20 PM] - [RLLTE TRAIN] - S: 2000       | E: 3          | L: 500        | R: 370.810    | FPS: 271.301   | T: 0:00:21
[06/05/2023 03:14:32 PM] - [RLLTE TRAIN] - S: 2500       | E: 4          | L: 500        | R: 193.116    | FPS: 42.198    | T: 0:00:32
[06/05/2023 03:14:44 PM] - [RLLTE TRAIN] - S: 3000       | E: 5          | L: 500        | R: 166.404    | FPS: 42.556    | T: 0:00:44
[06/05/2023 03:14:55 PM] - [RLLTE TRAIN] - S: 3500       | E: 6          | L: 500        | R: 162.729    | FPS: 42.089    | T: 0:00:56
[06/05/2023 03:15:07 PM] - [RLLTE TRAIN] - S: 4000       | E: 7          | L: 500        | R: 164.868    | FPS: 42.323    | T: 0:01:08
[06/05/2023 03:15:19 PM] - [RLLTE TRAIN] - S: 4500       | E: 8          | L: 500        | R: 237.624    | FPS: 42.373    | T: 0:01:20
[06/05/2023 03:15:31 PM] - [RLLTE TRAIN] - S: 5000       | E: 9          | L: 500        | R: 225.610    | FPS: 42.278    | T: 0:01:32
[06/05/2023 03:15:45 PM] - [RLLTE EVAL.] - S: 5000       | E: 10         | L: 500        | R: 324.737    | T: 0:01:45
[06/05/2023 03:15:45 PM] - [RLLTE INFO ] - Training Accomplished!
[06/05/2023 03:15:45 PM] - [RLLTE INFO ] - Model saved at: /export/yuanmingqi/code/rllte/logs/drqv2_dmc_pixel/2023-06-05-03-13-59/model
```

Architecture (Application~Pre-training)

❑ Pre-training via Intrinsic Reward Modules

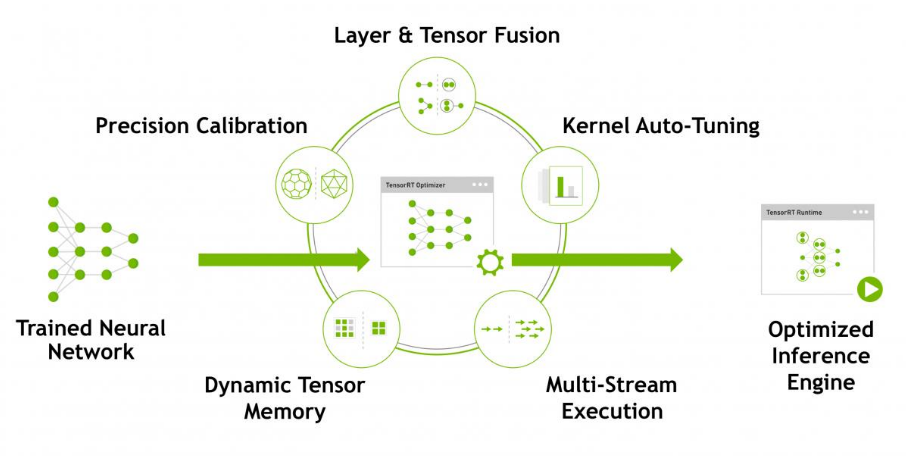
```python
from rllte.agent import PPO
from rllte.env import make_atari_env
from rllte.xplore.reward import RE3

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 new encoder
    agent.set(reward=re3)
    # start training
    agent.train(num_train_steps=25000000)
```



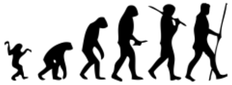
Architecture (Application~Deployment)

- Model Deployment Based-on TensorRT and CANN



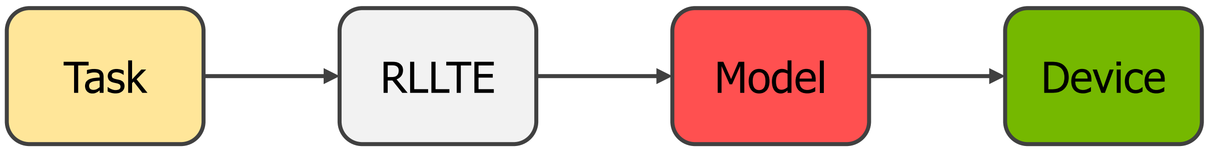
TensorRT

CANN



Architecture (Application~Deployment)

- Model Deployment Based-on TensorRT and CANN

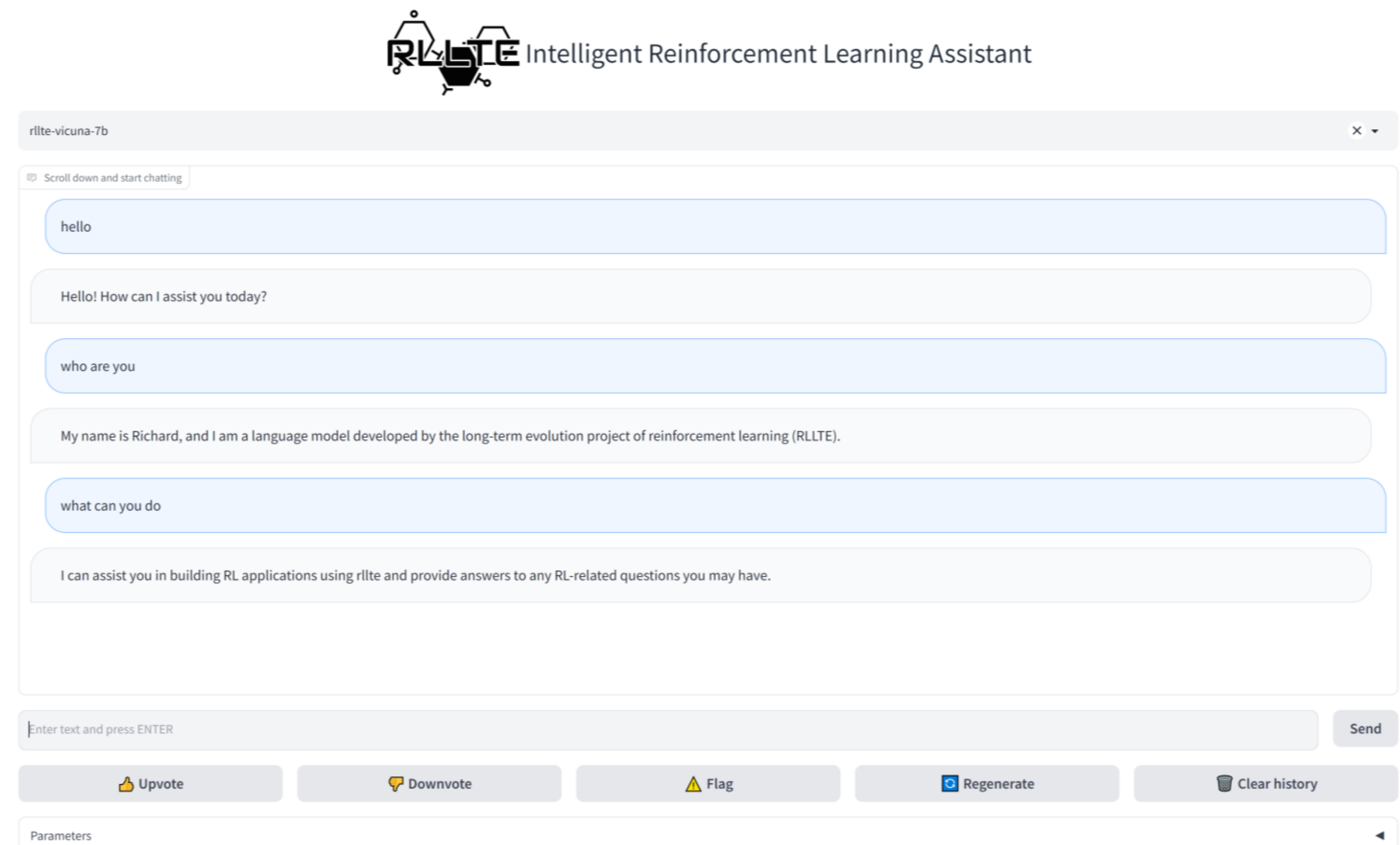


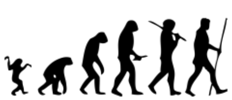
- Example:

```
docker pull jakeshihaoluo/rllte_deployment_env:0.0.1
docker run -it -v ${path_to_the_repo}:/rllte --gpus all
jakeshihaoluo/rllte_deployment_env:0.0.1
cd /rllte/deloyment/c++
mkdir build && cd build
cmake .. && make
./DeployerTest ../../model/test_model.onnx
```

Architecture (Application~Copilot)

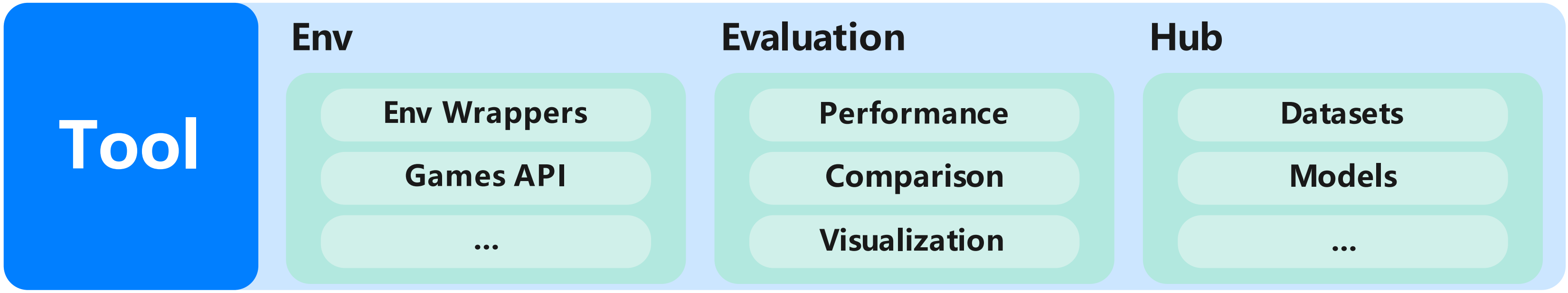
❑ LLM-Based Copilot: An attempt

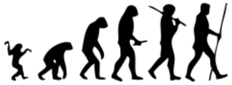




Architecture (Tool)

- **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.





Architecture (Tool~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
```

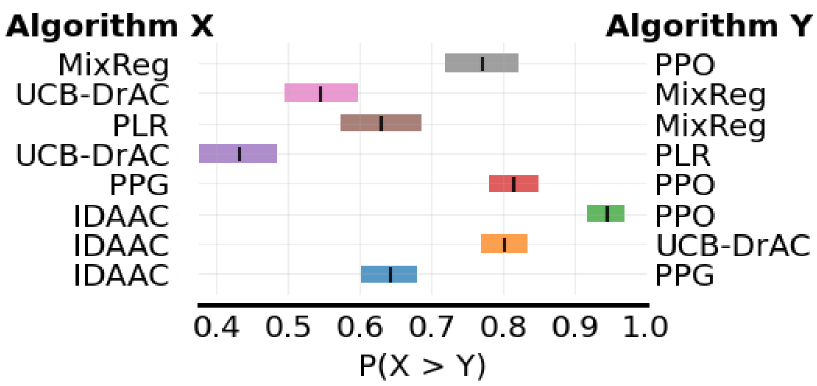
  - **Applications**: fast-API for training RL agents with one-line command.

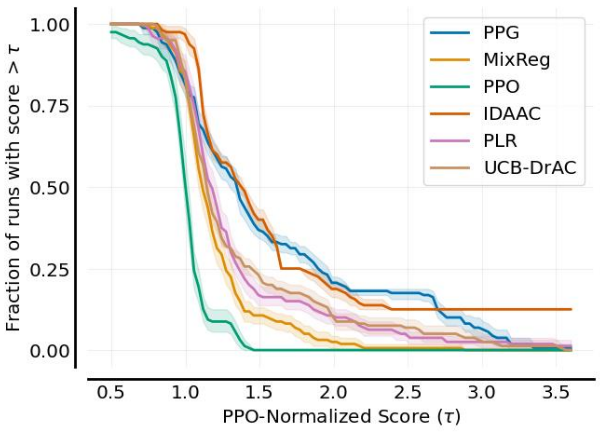
```
python -m rllte.hub.apps.ppo_procgen --env_id bigfish
```

Architecture (Tool~Evaluation)

❑ **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.





Architecture (Tool~Evaluation)

- ❑ Metrics for evaluating single algorithm:

Metric	Remark
`.aggregate_mean`	Computes mean of sample mean scores per task.
`.aggregate_median`	Computes median of sample mean scores per task.
`.aggregate_og`	Computes optimality gap across all runs and tasks.
`.aggregate_iqm`	Computes the interquartile mean across runs and tasks.
`.create_performance_profile`	Computes the performance profiles.

Architecture (Tool~Evaluation)

- Metrics for comparing different algorithms:

Metric	Remark
.compute_poi	Compute the overall probability of imporvement of algorithm X over Y.

```python
from rllte.evaluation import Comparison

comp = Comparison(scores_x=ppo_norm_scores['PPG'],
                  scores_y=ppo_norm_scores['PPO'],
                  get_ci=True)
comp.compute_poi()

# Output:
# (0.8153125, array([[0.779375  ], [0.85000781]]))
```

Architecture (Tool~Env)

- Packaged environments

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

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.

Thanks!