TorchRL

A data-driven decision-making library for PyTorch

Albert Bou, Matteo Bettini, Sebastian Dittert, Vikash Kumar, Shagun Sodhani, Xiaomeng Yang, Gianni De Fabritiis, Vincent Moens

TorchRL | TensorDict | Community | Paper

Abstract

TorchRL is an open source data-driven, general, decision-making library for PyTorch.

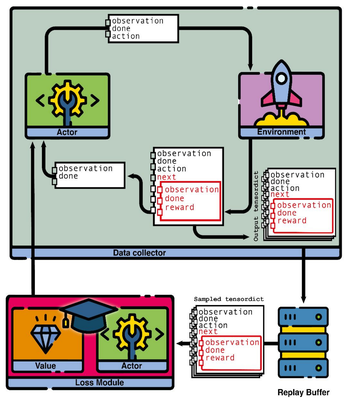
- Supports a **wide range of RL domains**: single-agent and multi-agent RL, online and offline RL, off-policy and on-policy RL, Model-free and Model-based RL.
- It is a **PyTorch domain library**, similar to TorchVision or TorchAudio, aiming to better support the RL community within the PyTorch ecosystem.

Motivation

Creating a truly general Reinforcement Learning (RL) library has historically proven very challenging due to several factors:

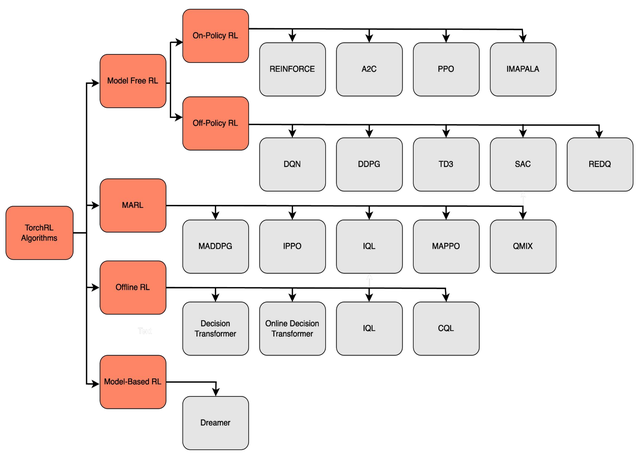
- **Algorithmic Complexity:** RL algorithms comprise numerous heterogeneous components that need to be combined.
- **Dynamic Data Requirements:** Components have varying input and output data requirements. Libraries are forced libraries to sacrifice flexibility to ensure good integration.
- **Specialized Use Cases:** Frameworks should accommodate specialized sub-domains (e.g. Offline RL, MARL) without redundancy.
- **Scaling Complexity:** Efficiently scaling poses greater challenges compared to supervised learning.
- **Long-Term Support:** Historically, frameworks have lacked sustained support, affecting viability over time.

TorchRL training logic example




```
# Environment
env = GymEnv("Pendulum-v1")
# Model: Actor and value
mlp_actor = MLP(num_cells=64, depth=3,
    in_features=3, out_features=1)
actor = TensorDictModule(mlp_actor,
    in_keys=["observation"], out_keys=["action"])
mlp_value = MLP(num_cells=64, depth=2,
    in_features=4, out_features=1)
critic = TensorDictSequential(actor,
    TensorDictModule(mlp_value,
    in_keys=["observation", "action"],
    out_keys=["state_action_value"]))
# Data Collector
collector = SyncDataCollector(
    env, AdditiveGaussianWrapper(actor),
    frames_per_batch=1000, total_frames=1000000)
# Replay Buffer
buffer = TensorDictReplayBuffer(
    storage=LazyTensorStorage(100000))
# Loss Module
loss_fn = DDPGClass(actor, critic)
optim = torch.optim.Adam(
    loss_fn.parameters(), lr=2e-4)
# Train
for data in collector:  # collect data
    buffer.extend(data)  # populate RB
    sample = buffer.sample(50)  # mini-batch
    loss = loss_fn(sample)  # loss
    loss = loss["loss_actor"] + loss["loss_value"]
    loss.backward()
    optim.step()
    optim.zero_grad()
```

TorchRL Algorithms



TorchRL Design Principles

TorchRL design principles tackle RL implementation challenges to keep PyTorch on the forefront of RL research and applications:

- **Standalone Components:** Low-level abstractions to solve independent, limited-scope RL problems.
- **Efficient Data Carrier:** Flexible and efficient communication between components, irrespective of their data requirements, with a new data carrier, the TensorDict.
- **Breadth over Depth:** Diversity of well-tested components to be used as the building blocks to cover a wide spectrum of RL sub-domains.
- **Minimal Core Dependencies:** primarily PyTorch and TensorDict.
- **Reliability and Long Term Support:** Within the PyTorch ecosystem, adhering to quality standards, ensuring maintenance.

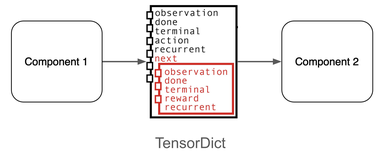
Powered by TensorDict

TensorDict is a **dictionary-like and tensor-like class**.

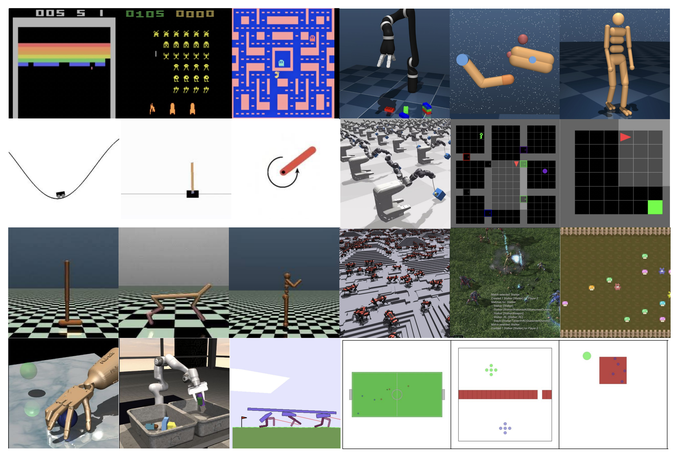
- Supports tensor operations like indexing, stacking, etc and point-to-point communication in distributed settings.
- Has support for non-tensor data, data serialisation and distributed capabilities.
- Transcends RL domain. It is already being used in projects outside this field.

TensorDict for RL component integration

- Key **data-structure powering** all of **TorchRL**.
- If all our components read and write to tensordicts, we create a system that is **agnostic to specific data signatures** and also allows for straightforward replacement of components with others to test different ideas.
- It also makes our code much readable, compact, and modular.



TorchRL Environments and Datasets



Environments

- Gym / Gymnasium
- dm_control
- Brax
- EnvPool
- Habitat
- Isaac Gym
- Jumanji
- Melting Pot
- OpenML
- Petting Zoo
- RoboHive
- StarCraft Multi-Agent Challenge v2
- Vectorized Multi-Agent Simulator (VMAS)

Datasets

- D4RL, VD4RL
- GenDGRL
- Roboset
- OpenX
- OpenML
- Minari
- AtariDQN

TorchRL ecosystem

- TorchRL has seen rapid growth since its initial release.
- The library has **more than 140 collaborators and contributors**.
- Has an active community on GitHub and Discord.
- **Over 20 applied research teams** from academia and industry have **adopted TorchRL** as a backend.
- The library features rich documentation, tutorials, a knowledge-base with RL insights, and state-of-the-art code implementations.