Gymnasium: Standard APIs for Reinforcement Learning

Aditya Patel

Why Standardization in RL Matters

- Before Gym (pre-2016): custom simulators, inconsistent resets, non-comparable results.
- Problem: no reproducibility; results from one lab could not be validated by others.
- Solution: a benchmark suite (shared set of environments + unified API).

Atari breakout

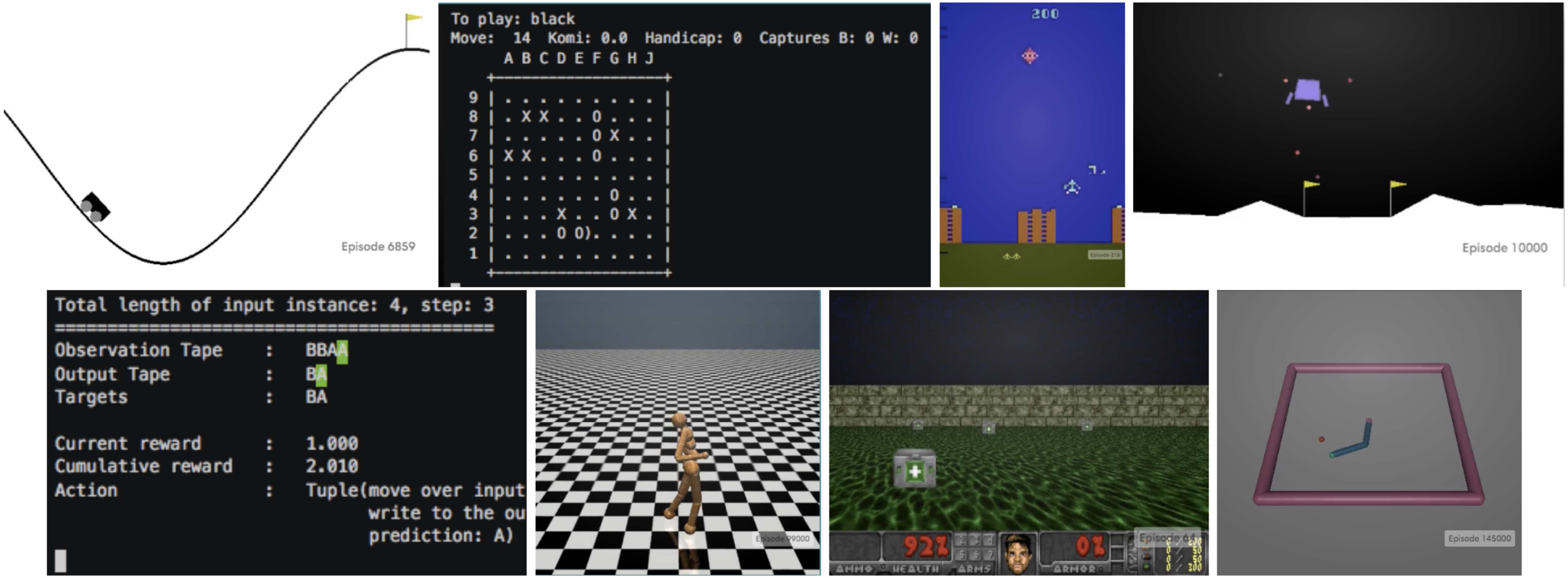
Why Standardization in RL Matters

- Before Gym (pre-2016): custom simulators, inconsistent resets, non-comparable results.
- Problem: no reproducibility; results from one lab could not be validated by others.
- Solution: a benchmark suite (shared set of environments + unified API).

Cannot compare athletes' speed if everyone is running on different terrain

Atari breakout

OpenAI Gym – Brockman, et al. 2016



Different environments in Gym

The Gym API: A Minimal Standard

Two key calls:

- reset() → initial observation.
- step(action) → returns (obs, reward, done, info).

Spaces: define legal actions & observations.

- Discrete(n)
- Box(low, high, shape)
- Dict, Tuple

Wrappers: transform environments without changing API

(e.g. stack frames, normalize rewards).

```
ob0 = env.reset() # sample environment state, return first observation
a0 = agent.act(ob0) # agent chooses first action
ob1, rew0, done0, info0 = env.step(a0) # environment returns observation,
# reward, and boolean flag indicating if the episode is complete.
a1 = agent.act(ob1)
ob2, rew1, done1, info1 = env.step(a1)
...
a99 = agent.act(o99)
ob100, rew99, done99, info2 = env.step(a99)
# done99 == True  =>  terminal
```

OpenAI - Gym 2016

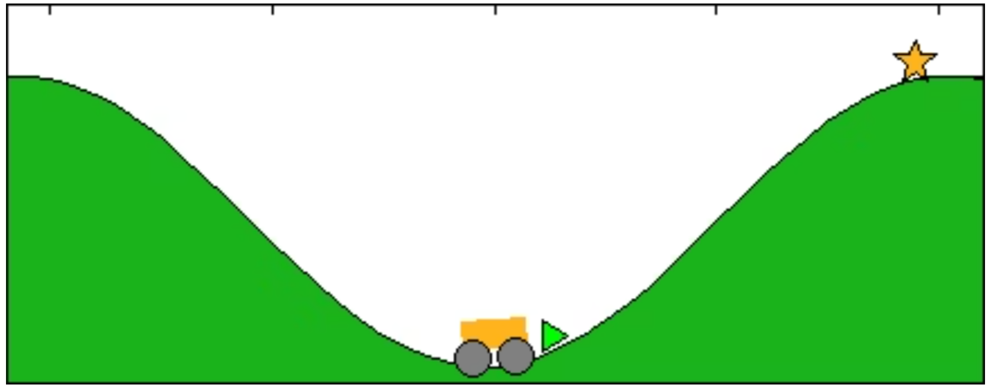
Gym's Impact and Ecosystem

Unified playground: Classic Control (CartPole, MountainCar), Atari (Breakout, Pong), MuJoCo (HalfCheetah, Ant).

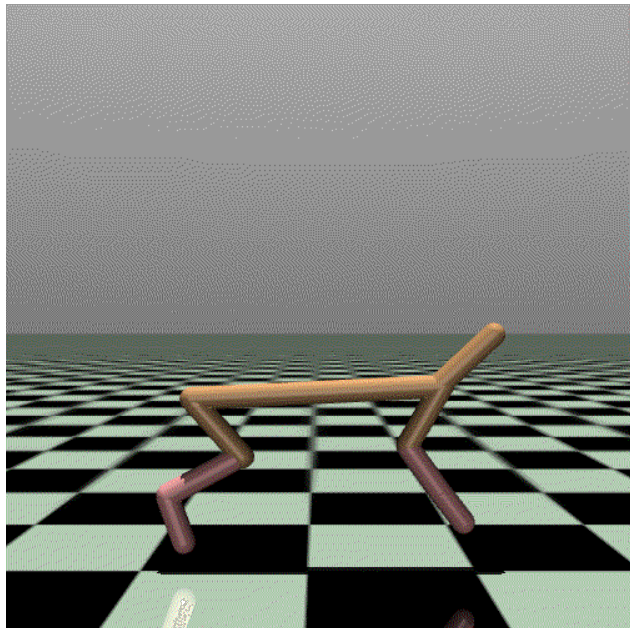
Enabled breakthrough algorithms:

- PPO (stable policy gradients),
- SAC (exploration via entropy),
- HER (goal relabeling in sparse rewards).

Integrates with RL libraries: Stable-Baselines3, RLlib, CleanRL, Dopamine.



MountainCar-v0, a simple environment emphasizing momentum and delayed reward.



HalfCheetah-v2 (MuJoCo), a continuous control agent learning to run efficiently.

MDP vs POMDP

Markov Decision Process (MDP)

- Agent has **full access to the true state**
- Observation = **state itself**
- Environment is **fully observable**
- Decision made **directly on current state**

Partially Observable MDP (POMDP)

- Agent has **limited or noisy access** to the stat
- Observation = **partial view** of the state
- Environment is **partially observable**
- Decision made **on a belief** (probability over states)

MDP vs POMDP

Markov Decision Process (MDP)

Full state observable: M = (S, A, T, R, µ)

- S is a set of states of the environment.
- A is a set of actions available to the agent.
- $T : S \times A \rightarrow \Delta S$ is the environment transition function, representing its dynamics.
- $R: S \times A \times S \rightarrow R$ is the reward function which is used to define the agent's task.
- $\mu \in \Delta S$ is the initial state distribution

Partially Observable MDP (POMDP)

Partial observation: M = (S, A, T, R, Ω, O, µ)

- S is a set of states of the environment.
- A is a set of actions available to the agent.
- $T : S \times A \rightarrow \Delta S$ is the environment transition function, representing its dynamics.
- $R: S \times A \times S \rightarrow R$ is the reward function which is used to define the agent's task.
- Ω is a set of possible observations
- $O: S \rightarrow \Delta\Omega$ is the observation function mapping states to observations.
- $\mu \in \Delta S$ is the initial state distribution

Reason for Decline

Ambiguous termination: conflated task failure (true terminal) and timeouts (artificial cutoff).

Leads to wrong value estimation.

Not theory aligned: API didn't reflect POMDP structure.

Vectorization was an afterthought: inefficient parallel execution.

Stalled maintenance: API Maintainace stopped after 2021

Gymnasium – Towers et al. 2024

01

Created by the Farama Foundation (2024) as successor to Gym.

02

Backward compatible

03

Clearer semantics: aligns with POMDP definitions.

04

Practical improvements: explicit episode endings, reproducibility guarantees, scalable vectorization.

Gymnasium Core Abstractions

Env: single task, refined API.

VectorEnv: formalizes parallel stepping (sync/async/custom).

Space: extended to model structured data (Dict, Graph, OneOf).

Registry: strict versioning, reproducible experiments.

Novel Idea -Functional API (FuncEnv): POMDP-Aligned

initial → start state distribution.

observation → what agent sees.

transition → next state given action.

reward → feedback function.

terminal → end condition. (doesn’t correspond to POMDP theory, but super important)

```
s = reset_fn(key)            # sample initial state
o = observation_fn(s)          # get initial observation

a = policy(o)                          # agent chooses action
 s_next = transition_fn(s,a)           # next state from dynamics T(s,a)
 r = reward_fn(s,a,s_next)             # compute reward R(s,a,s′)
 done = terminal_fn(s_next)            # true terminal condition
 trunc = truncation_fn(t)              # time or boundary cutoff
 o = observation_fn(s_next)            # next observation
 s = s_next                            # update state
 if done or trunc:                     # stop if ended or truncated
     break
```

Matches formal POMDP tuple (S,A,T,R,O,μ).

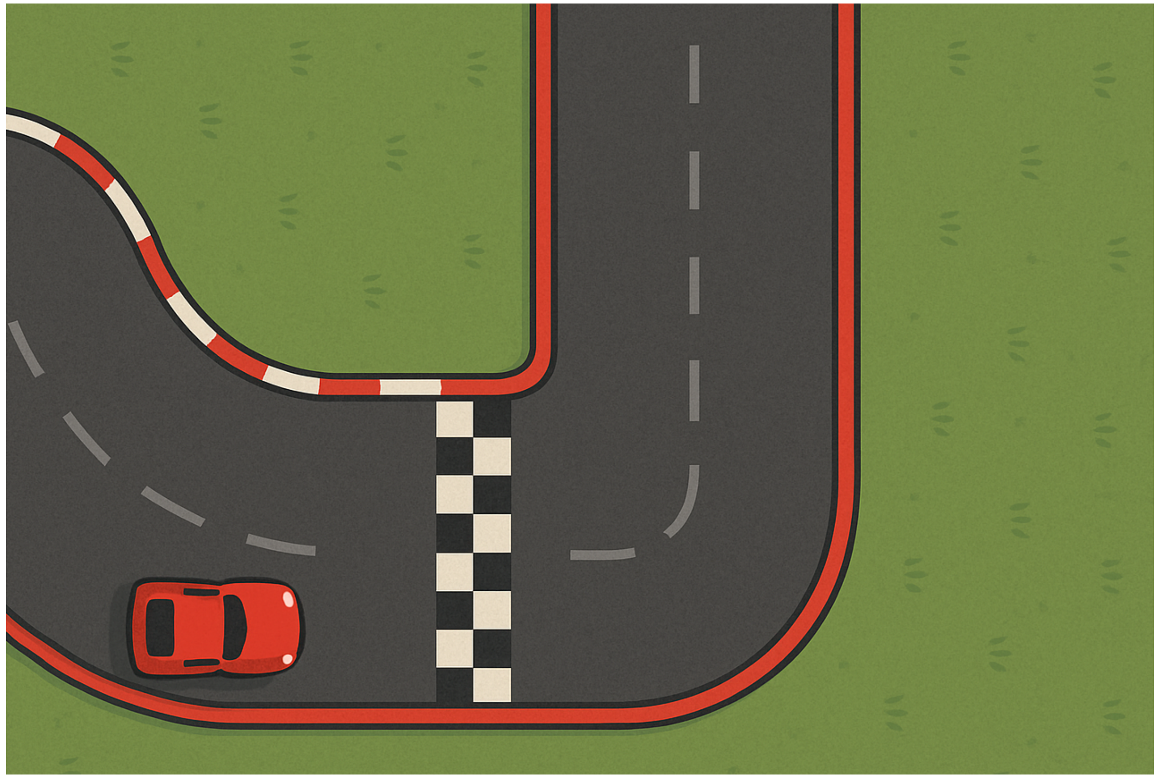
Functional design = JAX/NumPy friendly; efficient batching, reproducible.

Episode Endings: Termination vs Truncation

- **Terminated**: true end of task. E.g., pole falls over.
- **Truncated**: cutoff imposed by time/budget. E.g., simulation stops after 200 steps even though pole is balanced.

This matters because of bootstrapping:

- At termination, future rewards = 0.
- At truncation, the next-state still has potential future value.



Algebraic Spaces: Richer Modeling

1. Beyond simple Box and Discrete.

2. Composite and structured types:

   Dict: multimodal observations (e.g. images + proprioception).

   Sequence: variable-length inputs (e.g. NLP).

   Graph: relational structures (e.g. social networks, molecule environments).

   OneOf: actions that could be discrete or continuous.

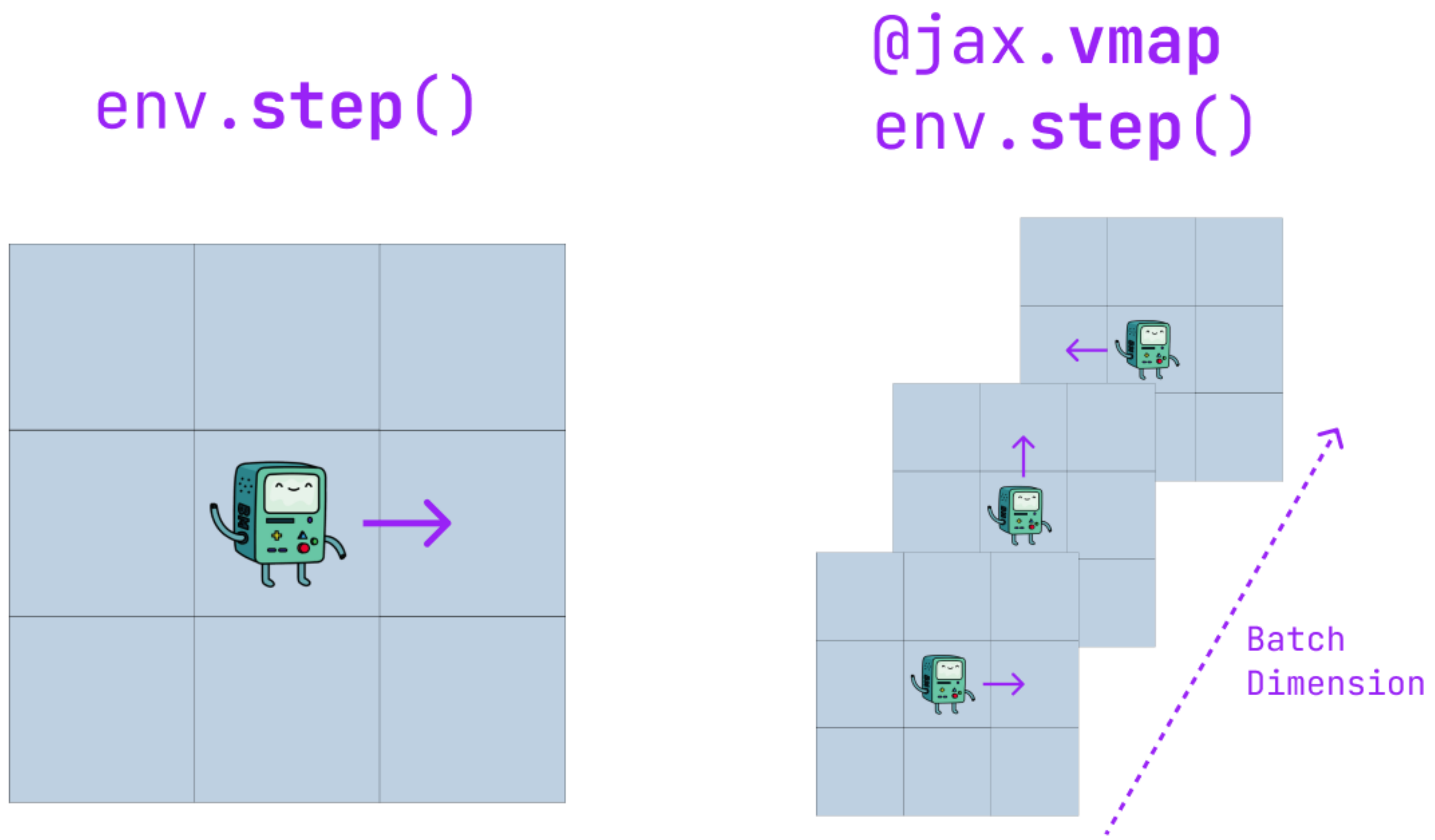
3. Mirrors algebraic data types → more natural modeling.

Algebraic Spaces: Richer Modeling

```python
# Original env: torque control (in Newton-meters)
env = gym.make("PandaReach-v3")
original_space = env.action_space  # e.g., Box(-10, 10, shape=(3,))

# Transform: normalize torques to [-1, 1] for the policy
normalized_space = original_space.map(
    lambda x: Box(low=-1.0, high=1.0, shape=x.shape)
)

# Agent acts in normalized space
action_norm = normalized_space.sample()          # e.g., [0.5, -0.2, 0.1]
action_real = np.interp(action_norm, [-1, 1],    # Map back before applying
                        [original_space.low, original_space.high])

obs, reward, done, _, _ = env.step(action_real)
```

Built-in Vectorization

- Vectorization enables performance gains without major changes to algorithm implementation
- SyncVectorEnv: runs N envs sequentially and batches results, fast for lightweight tasks
- AsyncVectorEnv: runs envs in subprocesses, better for heavier tasks.
- Practical impact: RL training loops much faster when many envs can run in parallel.
- Is the choice arbitrary?



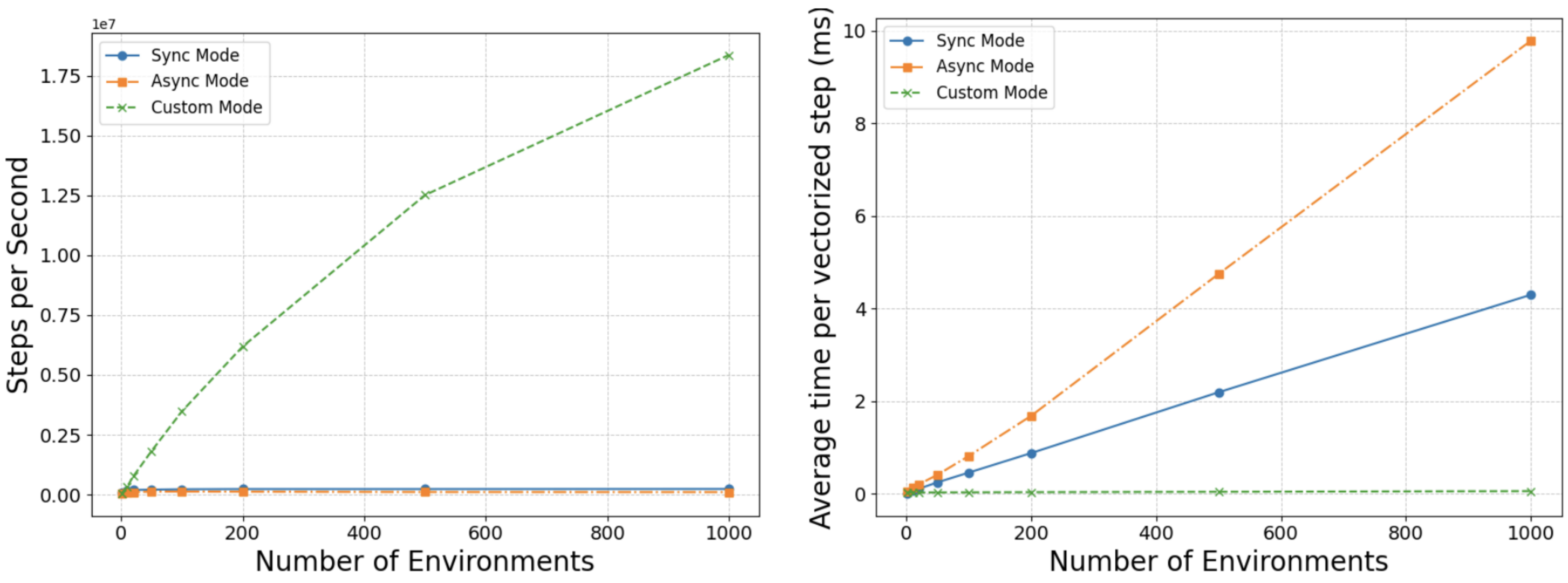
Single environment,
single agent step

Multiple environments,
Multiple agents steps

https://medium.com/data-science/vectorize-and-parallelize-rl-environments-with-jax-q-learning-at-the-speed-of-light-49d07373adf5

Vectorization

- Simple & lightweight Cartpole Game
- Custom mode utilizes numpy based vectorization.
- Sync mode outperforms Async
- Custom mode outperforms both
- Async is overkill

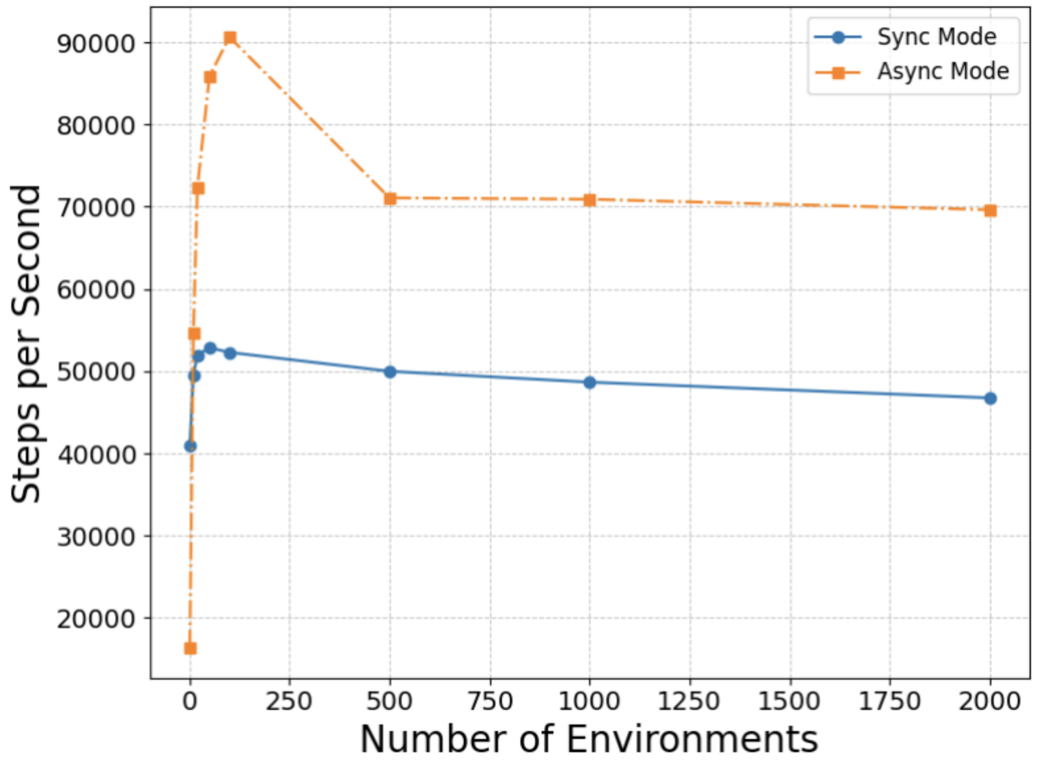


(a) Individual environment steps per second. Higher is better.

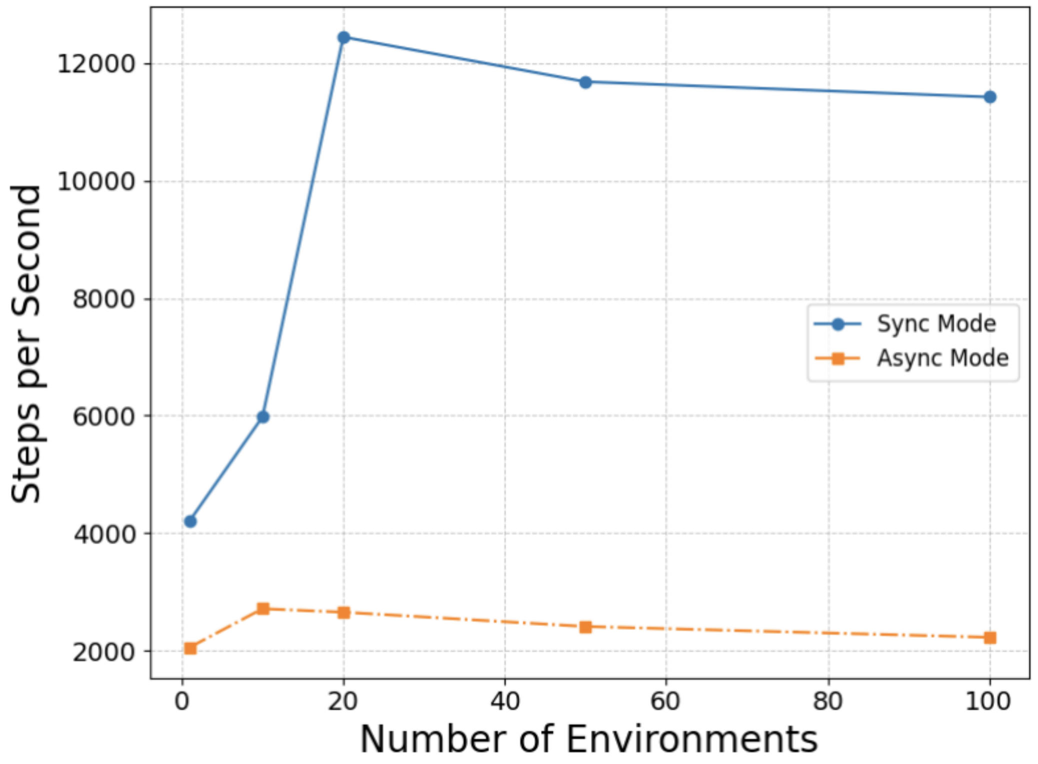
(b) Average time to perform a single batched step. Lower is better.

Vectorization

- Conversely, in a more complex env like lunar lander:



(a) Individual environment steps per second, executed on a MacBook Pro. Higher is better.



(b) Individual environment steps per second, executed on Google Colab. Higher is better.

Environments and Ecosystem

- Core suites: Toy Text, Classic Control, Box2D, MuJoCo (continuity with Gym).