license MIT | Models Huggingface | Weights & Biases benchmark | code style black | imports isort

SCAN ME

Cleanba: A Reproducible and Efficient Distributed Reinforcement Learning Platform

Repo: https://github.com/vwxyzjn/cleanba

Shengyi Huang, Jiayi Weng, Rujikorn Charakorn, Min Lin, Zhongwen Xu, Santiago Ontanon

Drexel University, Hugging Face
Google, VISTEC, Sea AI Lab
Tencent AI Lab

TL;DR

- 📜 **Single-file PPO and IMPALA implementation**
  — Each file is about 700 LOC (not counting evaluation code)
- ✅ **Benchmarked implementation**
  — Evaluated on 57 Atari games, outperformed Moolib and Monobeast's IMPALA
- 📈 **Highly reproducible**
  — Reproducible across different hardware settings
- 😈 **Implemented with JAX and EnvPool**
  — Highly efficient; scalable to hundreds of GPUs

IMPALA's reproducibility issues

- 🤔 **In IMPALA, what happens if the learner updates while the actor is in mid-rollout?**

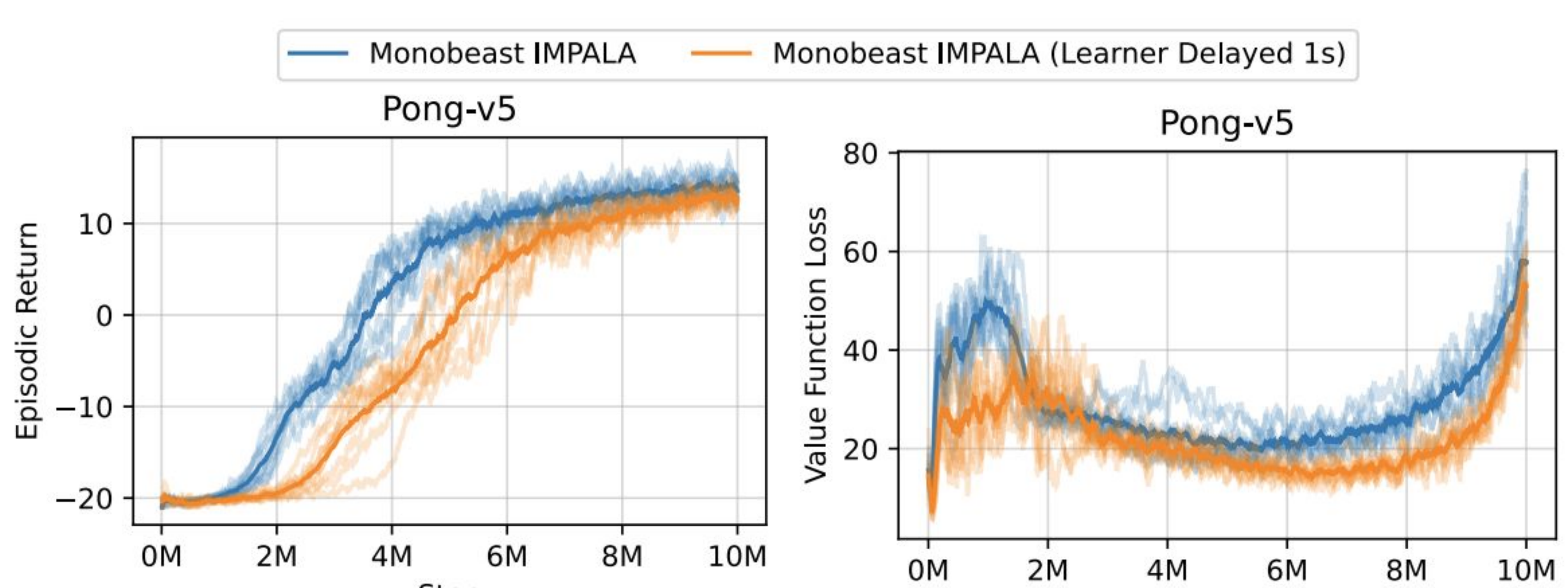
IMPALA Actor-Learner Architecture

```
batch_size = 32
agent = Agent()
data_Q = queue()

def actor():
  while True:
    data = rollout(agent.param, 1)

    data_Q.put(data)
def learner():
  for _ in range(1, ITER):
    data = data_Q.get_many(batch_size)
    agent.learn(data)
    broadcast_to_actors(agent.param)
for _ in range(num_actors):
  thread(actor).start()
thread(learner).start()
```

Cleanba's architecture

```
batch_size = 32
agent = Agent()
data_Q = queue(max_size=1)
param_Q = queue(max_size=1)
def actor():
  for i in range(1, ITER):
    if i != 2:
      params = param_Q.get()
    data = rollout(params, batch_size)
    data_Q.put(data)
def learner():
  for _ in range(1, ITER):
    data = data_Q.get()
    agent.learn(data)
    param_Q.put(agent.param)
param_Q.put(agent.param)
thread(actor).start()
thread(learner).start()
```

- 🕵️ **The policies that create learner trajectories are non-deterministic: same hyperparameters could yield unreproducible learning curves**



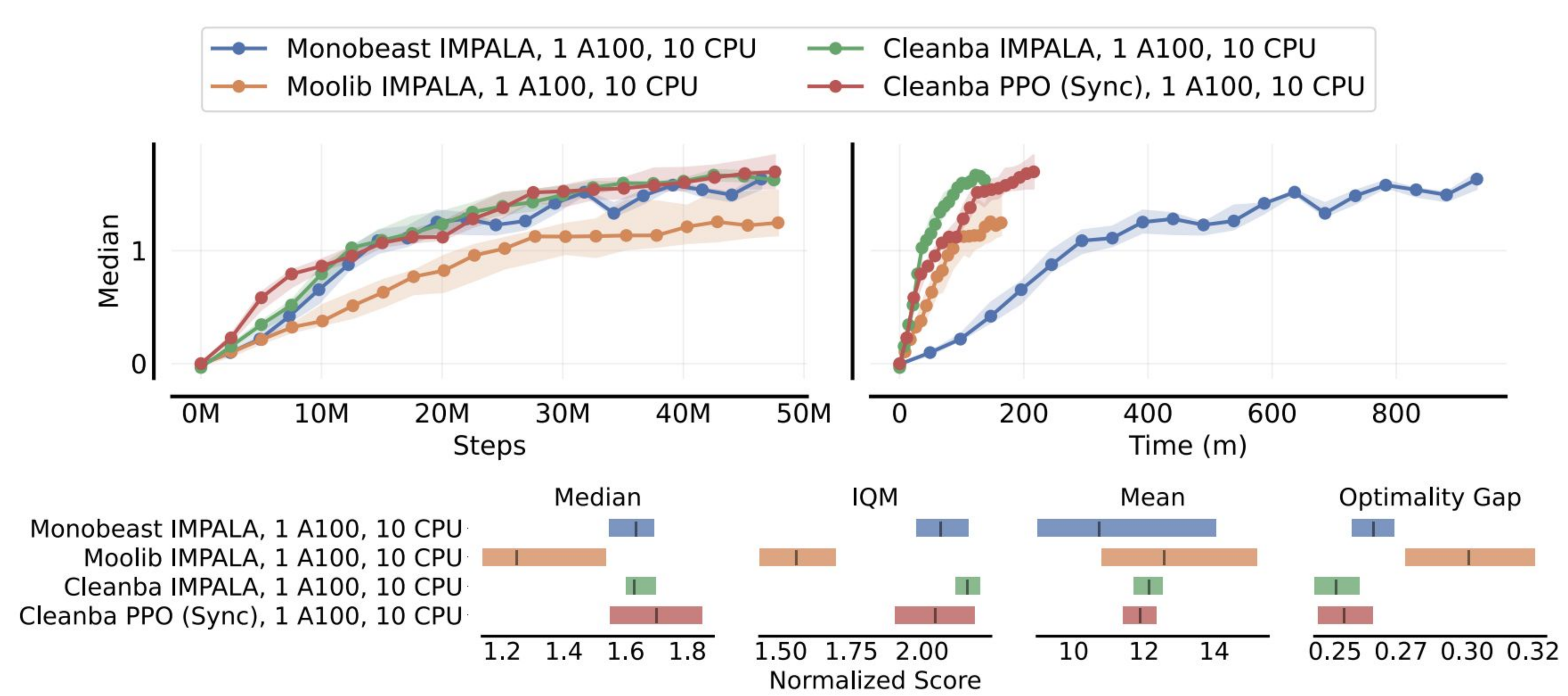
Cleanba's architecture

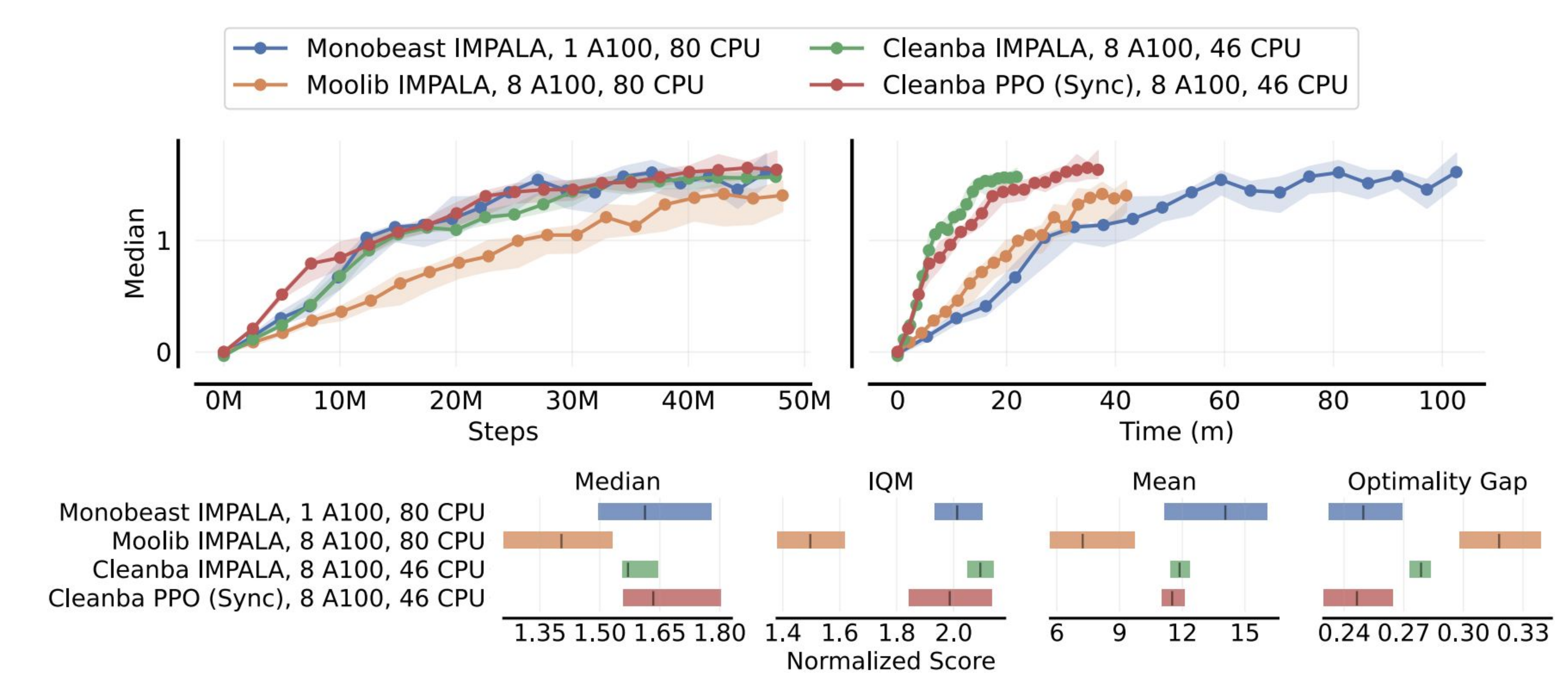
- 👨‍🔬 **Simple idea to ensure reproducibility; always learn from the second latest policy**

Iteration	1	2	3
Synchronous Arch.	$\pi_1 \to \mathcal{D}_{\pi_1}$ $\pi_1 \xrightarrow[\mathcal{D}_{\pi_1}]{} \pi_2$	$\pi_2 \to \mathcal{D}_{\pi_2}$ $\pi_2 \xrightarrow[\mathcal{D}_{\pi_2}]{} \pi_3$	$\pi_3 \to \mathcal{D}_{\pi_3}$ $\pi_3 \xrightarrow[\mathcal{D}_{\pi_3}]{} \pi_4$
Cleanba's Arch., Actor	$\pi_1 \to \mathcal{D}_{\pi_1}$	$\pi_1 \to \mathcal{D}_{\pi_1}$	$\pi_2 \to \mathcal{D}_{\pi_2}$
Cleanba's Arch., Learner		$\pi_1 \xrightarrow[\mathcal{D}_{\pi_1}]{} \pi_2$	$\pi_2 \xrightarrow[\mathcal{D}_{\pi_1}]{} \pi_3$

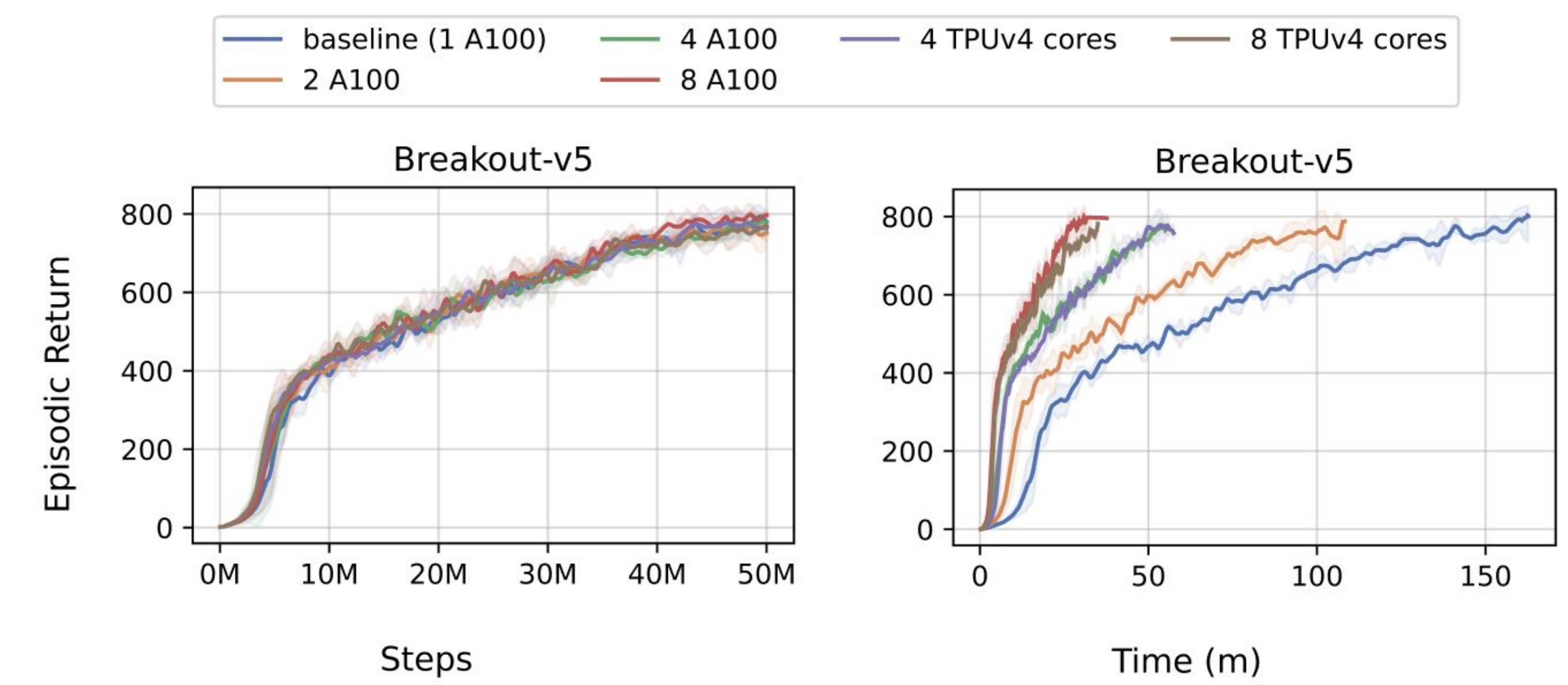
Evaluation

- 🔥 **Outperforms moolib and monobeast's IMPALA**
  — Same Atari wrappers (e.g., sticky actions)
  — Base hardware setting (1 A100, 10 CPU) and Workstation setting.
  — Using rliable for plotting median, IQM, mean, and optimality gap

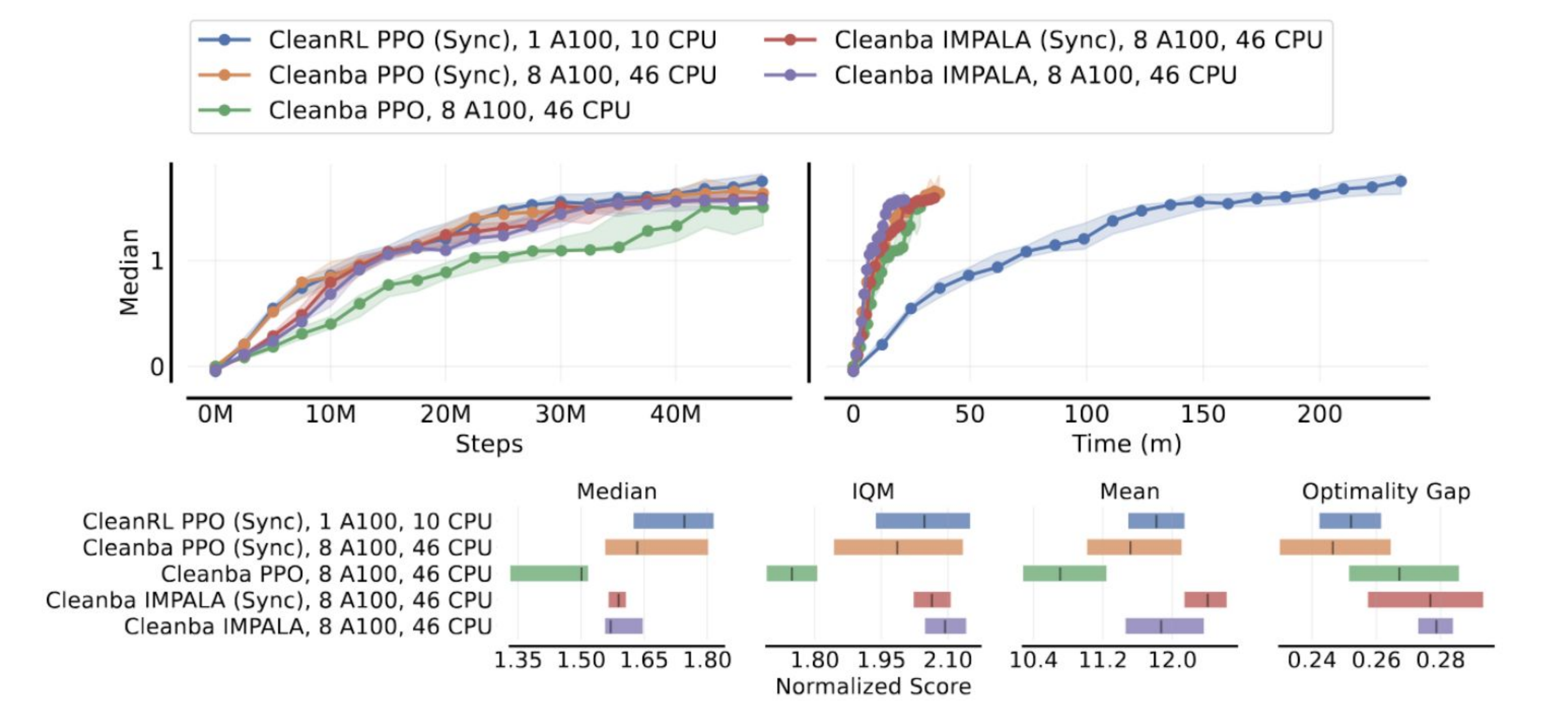




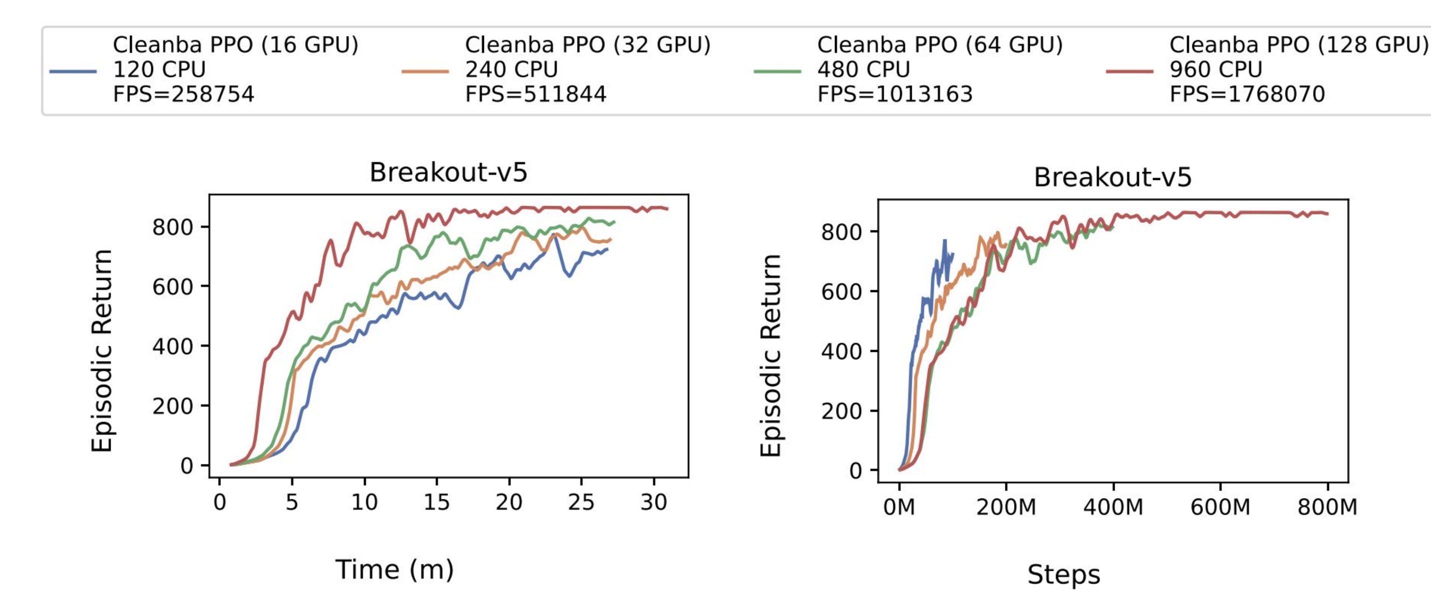
- 🔨 **Reproducible across different hardwares**
  — Tested with different number of GPUs and TPUs
  — Identical learning curves; just different runtimes



- 👀 **To sync or not to sync**
  — PPO (sync) is slower but has high data efficiency
  — IMPALA is faster than IMPALA (sync); same data efficiency



- 🚀 **Cleanba PPO can scale to hundreds of GPUs**



Acknowledgment

- 🙏 **We thank Stability AI's HPC, Hugging Face's cluster, and Google's TPU Research Cloud for computes**