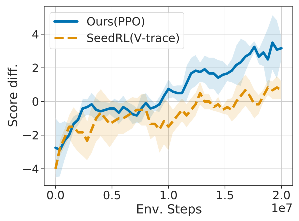

Figure 1: 11 vs. 11 easy.

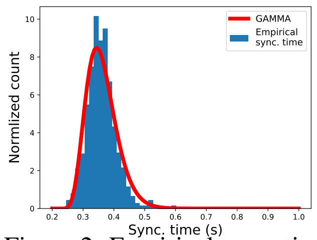

Figure 2: Empirical sync. time.

Method	Kostrikov	OpenAI Baselines	rlpyt	Ours
BankHeist	1382 ±6	991 ±14	1737 ±39	**2111 ±21**
Beam Rider	1663 ±14	1081 ±18	2086 ±32	**2586 ±14**
Breakout	1225 ±12	829 ±31	1508 ±60	**1885 ±15**
Frostbite	1337 ±8	962 ±15	1803 ±17	**1973 ±24**
Jamesbond	1353 ±5	1014 ±1	1991 ±24	**2139 ±31**
Krull	1443 ±6	1057 ±11	2001 ±29	**2657 ±16**
KFMaster	1532 ±15	1056 ±8	1979 ±55	**2483 ±15**
MsPacman	1574 ±9	1052 ±3	1972 ±13	**2364 ±5**
Qbert	1232 ±13	953 ±7	1621 ±43	**1860 ±6**
Seaquest	1593 ±10	946 ±21	1918 ±25	**2633 ±32**
S. Invader	1514 ±20	1010 ±7	1899 ±32	**2318 ±12**
Star Gunner	1622 ±19	1110 ±5	2066 ±24	**2616 ±25**

Table 1: SPS of different implementations of A2C.

General response: We thank all reviewers for their comments.

Reviewer # 1:

Q1: Comparison with Seed RL. We compare HTS-RL with Seed RL (V-trace) on Gfootball '11vs.11 easy'. For a fair comparison, both HTS-RL and Seed RL use 16 parallel environment processes and one GPU. HTS-RL achieves 829 environment steps per second (SPS) while Seed RL achieves 609 SPS. After 20M steps of training, HTS-RL and Seed RL achieve a 3.55 ± 0.3 and 1.50 ± 0.7 score difference, respectively. The training curve is shown in Fig. 1.

Q2: Environment step time assumption. In Claim 1, we assume the step times follow an exponential distribution. Hence the sum of step times (synchronization time) follows a Gamma distribution. We empirically verify this assumption. In Fig. 2, we show the histogram of synchronization time (sum of every 100 step times) on 'academy 3vs1 w/ keeper'. Furthermore, we perform a Kolmogorov-Smirnov goodness-of-fit test, with a significance-level of 0.05 and D-statistics of 0.04. We find the empirical data is consistent with the assumed Gamma distribution.

Reviewer # 2:

Q3: Scientific contribution of this work. Like all published high-throughput RL frameworks, *e.g.*, A3C, GA3C, IMPALA, and SeedRL, the scientific contribution of HTS-RL is the reduction in training time. This scales RL research to more complex tasks which is important. Also, HTS-RL maintains the advantages of synchronous RL, *i.e.*, data efficiency and reproducibility, while achieving speedups. Additionally, we provide a detailed analysis of the proposed technique. All reviewers agree that practitioners, *i.e.*, researchers, in our community could benefit from HTS-RL.

Q4: Placement of figures. We'll rearrange the figures.

Reviewer # 3:

Q5: Compared Baselines & Additional comparisons with OpenAI and rlpyt. We compare the speed of different versions of A2C, including our HTS-RL, Kostrikov, OpenAI baselines, and rlpyt, on Atari games. For a fair comparison, all methods use 16 parallel environment processes for data collection, and one GPU for model training/forwarding. For rlpyt, we use the most efficient 'parallel-GPU' mode. As shown in Tab. 1, Kostrikov's A2C is a strong baseline, which achieves 1.4× higher SPS than OpenAI baselines. Also, HTS-RL consistently achieves higher SPS than rlpyt.

Q6: Pseudo code for clarity. Actual code is part of this submission. We'll add pseudo code.

Q7: Practicality of *i.i.d.* assumption of environment step time in Claim 1. We think an *i.i.d.* environment step time assumption is practical for a machine which satisfies the following conditions: 1) CPU cores are homogeneous; and 2) a CPU's speed is not influenced by the workload of other CPUs. Because we run the same environment on homogeneous CPUs (condition (1)), the step times are identically distributed. In addition, each environment doesn't intervene with any other environments, *i.e.*, environments are independent. From condition (2), the speed of each CPU is also independent. Therefore, the step time in each environment is independent.

Q8: Batch synchronization may not work in some extreme cases. In the extreme case described by the reviewer (one executor is 10× slower than the other), we agree that any kind of synchronization decreases throughput. However, empirically, we never observed such cases.

Q9: # of environments vs. # of threads. In all experiments, each environment runs independently in a single-threaded process. *E.g.*, in Kostrikov's A2C with 16 environments, 16 envs are handled by 16 parallel single-threaded processes. We do not run multiple environments in one thread for any method. We'll clarify use of 'thread' and 'process.'

Q10: Crediting double replay buffer and batch synchronization. To our understanding, the sampling schemes in rlpyt differ from our 'batched synchronization.' One may confuse the 'parallel-GPU' sampling scheme of rlpyt with 'batch synchronization.' However, rlpyt's 'parallel-GPU' mode enforces synchronization across workers at **every** environment batch-step. In contrast, batched synchronization synchronizes all environments every α ($\alpha > 1$) steps (L.187-193). rlpyt uses the double replay buffer in asynchronous mode to increases throughput [A]. In contrast, as discussed in L.181-186, we use a double replay buffer to avoid a stale policy and to maintain determinism in synchronous training. We'll clarify.

References: [A] Stooke et al. rlpyt: A Research Code Base for Deep Reinforcement Learning in PyTorch, arxiv. 2019

Reviewer # 4:

Q11: Scientific contribution of this work. Please see Q3.

Q12: Addressing the off-policy updates? We address the off-policy update in Sec. 4.1 (L.202-215) via a one-step-delayed gradient. In addition, in Appendix C, we show that under assumptions, one-step-delayed gradient has the same asymptotic convergence rate as the zero-delayed case.

Q13: Experiment demonstrating determinism. Please see Tab. A1 and L.474-475 in the appendix. Even with a different number of actors, the final average scores obtained by HTS-RL are exactly identical. This demonstrates the deterministic nature of HTS-RL.