IMPACT: IMPORTANCE WEIGHTED ASYNCHRONOUS ARCHITECTURES WITH CLIPPED TARGET NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

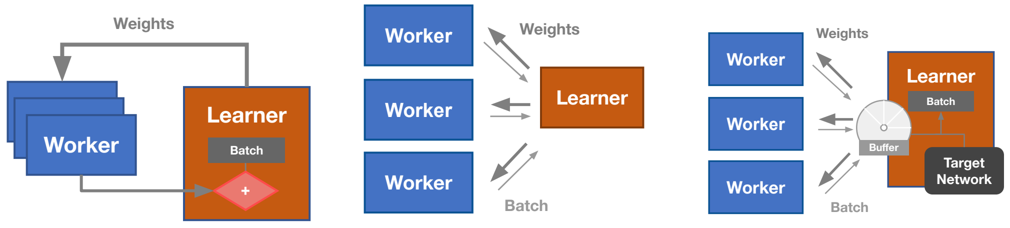
The practical usage of reinforcement learning agents is often bottlenecked by the duration of training time. To accelerate training, practitioners often turn to distributed reinforcement learning architectures to parallelize and accelerate the training process. However, modern methods for scalable reinforcement learning (RL) often tradeoff between the throughput of samples that an RL agent can learn from (sample throughput) and the quality of learning from each sample (sample efficiency). In these scalable RL architectures, as one increases sample throughput (i.e. increasing parallelization in IMPALA (Espeholt et al., 2018)), sample efficiency drops significantly. To address this, we propose a new distributed reinforcement learning algorithm, IMPACT. IMPACT extends PPO with three changes: a target network for stabilizing the surrogate objective, a circular buffer, and truncated importance sampling. In discrete action-space environments, we show that IMPACT attains higher reward and, simultaneously, achieves up to 30% decrease in training wall-time than that of IMPALA. For continuous control environments, IMPACT trains faster than existing scalable agents while *preserving the sample efficiency of synchronous PPO*.

1 INTRODUCTION

Proximal Policy Optimization (Schulman et al., 2017) is one of the most sample-efficient on-policy algorithms. However, it relies on a synchronous architecture for collecting experiences, which is closely tied to its trust region optimization objective. Other architectures such as IMPALA can achieve much higher throughputs due to the asynchronous collection of samples from workers. Yet, IMPALA suffers from reduced sample efficiency since it cannot safely take multiple SGD steps per batch as PPO can. The new agent, **Imp**ortance Weighted **A**synchronous Architectures with **C**lipped **T**arget Networks (**IMPACT**), mitigates this inherent mismatch. Not only is the algorithm highly sample efficient, it can learn quickly, training 30 percent faster than IMPALA. At the same time, we propose a novel method to stabilize agents in distributed asynchronous setups and, through our ablation studies, show how the agent can learn in both a time and sample efficient manner.

In our paper, we show that the algorithm IMPACT realizes greater gains by striking the balance between high sample throughput and sample efficiency. In our experiments, we demonstrate in the experiments that IMPACT exceeds state-of-the-art agents in training time (with same hardware) while maintaining similar sample efficiency with PPO. The contributions of this paper are as follows:

1. We show that when collecting experiences asynchronously, introducing a target network allows for a stabilized surrogate objective and multiple SGD steps per batch (Section 3.1).
2. We show that using a circular buffer for storing asynchronously collected experiences allows for smooth trade-off between real-time performance and sample efficiency (Section 3.2).
3. We show that IMPACT, when evaluated using identical hardware and neural network models, improves both in real-time and timestep efficiency over both synchronous PPO and IMPALA (Section 4).



(a) PPO (b) IMPALA (c) IMPACT

Figure 1: Architecture schemes for distributed PPO, IMPALA, and IMPACT. PPO aggregates worker batches into a large training batch and the learner performs minibatch SGD. IMPALA workers asynchronously generate data. IMPACT consists of a batch buffer that takes in worker experience and a target's evaluation on the experience. The learner samples from the buffer.

2 Background

Reinforcement Learning assumes a Markov Decision Process (MDP) setup defined by the tuple (S, A, p, γ, r) where S and A represent the state and action space, $\gamma \in [0, 1]$ is the discount factor, and $p : S \times A \times S \to \mathbb{R}$ and $R : S \times A \to \mathbb{R}$ are the transition dynamics and reward function that models an environment.

Let $\pi(a_t|s_t) : S \times A \to [0, 1]$ denote a stochastic policy mapping that returns an action distribution given state $s_t \in S$. Acting out with $\pi(a_t|s_t)$ is equivalent to sampling a trajectory $\tau \sim \mathbb{P}(\boldsymbol{\tau}|\pi)$, where $\tau := (s_0, a_0, ..., a_{T-1}, s_T, a_T)$. We can compactly define the trajectory distribution $\mathbb{P}(\boldsymbol{\tau}|\pi)$ with state and state-action marginals $p_\pi(s_t)$ and $p_\pi(s_t, a_t)$.The goal for reinforcement learning aims to maximize the follow objective: $J(\cdot) = \mathbb{E}_{(s_t,a_t)\sim p_\pi}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$.

When θ parameterizes $\pi(a_t|s_t)$, performing a gradient w.r.t θ is equivalent to the Policy Gradient Theorem (Sutton & Barto, 2018):

$$\nabla_\theta J(\theta) = \mathbb{E}_{(s_t,a_t)\sim p_\pi(\cdot)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_{\pi_\theta}(s_t, a_t)\right],$$

where $\hat{A}_{\pi_\theta}(s_t, a_t)$ is an estimator of the advantage function. Policy gradients, however, suffer from high variance and large update-step sizes, oftentimes leading to sudden drops in performance.

2.1 Distributed PPO

Per iteration, Proximal Policy Optimization (PPO) optimizes policy π_θ from target $\pi_{\theta_{\text{old}}}$ via the following objective function

$$L(\theta) = \mathbb{E}_{p_{\pi_{\theta_{\text{old}}}}} \left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\right)\hat{A}_t\right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ and ϵ is the clipping hyperparameter. In addition, many PPO implementations use GAE-λ as a low bias, low variance advantage estimator for $\hat{A}_t$ (Schulman et al., 2015b). PPO's surrogate objective contains the importance sampling ratio $r_t(\theta)$, which can potentially explode if $\pi_{\theta_{\text{old}}}$ is too far from π_θ. (Han & Sung, 2017). PPO's surrogate loss mitigates this with the clipping function, which ensures that the agent makes reasonable steps. Alternatively, PPO can also be seen as an adaptive trust region introduced in TRPO (Schulman et al., 2015a).

In Figure 1a, distributed PPO agents implement a synchronous data-gathering scheme. Before data collection, workers are updated to π_{old} and aggregate worker batches to training batch D_{train}. The learner performs many mini-batch gradient steps on D_{train}. Once the learner is done, learner weights are broadcast to all workers, who start sampling again.

2.2 Importance Weighted Actor-Learner Architectures

In Figure 1b, IMPALA decouples acting and learning by having the learner threads send actions, observations, and values while the master thread computes and applies the gradients from a queue of learners experience (Espeholt et al., 2018). This maximizes GPU utilization and allows for increased sample throughput, leading to high training speeds on easier environments such as Pong. As the number of learners grows, worker policies begin to diverge from the learner policy, resulting in stale policy gradients. To correct this, the IMPALA paper utilizes V-trace to correct the distributional shift:

$$v_{s_t} = V_\phi(s_t) + \sum_{i=t}^{t+n-1} \gamma^{i-t} \left(\prod_{j=t}^{i-1} c_j \right) \rho_i \left(r_{i+1} + \gamma V_\phi(s_{i+1}) - V_\phi(s_i) \right)$$

where, V_ϕ is the value network, π_θ is the policy network of the master thread, $\mu_{\theta'}$ is the policy network of the learner thread, and $c_i = \rho_i = \left(\frac{\pi_\theta(a_i|s_i)}{\mu_{\theta'}(a_i|s_i)} \right)$.

Algorithm 1 IMPACT

```
Input: Batch size $M$, number of workers $W$, circular buffer size $N$, replay coefficient $K$, target
    update frequency $t_{\text{target}}$, weight broadcast frequency $t_{\text{frequency}}$, learning rates $\alpha$ and $\beta$
 1: Randomly initialize network weights $(\theta, w)$
 2: Initialize target network $(\theta', w') \leftarrow (\theta, w)$
 3: Create $W$ workers and duplicate $(\theta, w)$ to each worker
 4: Initialize circular buffer $C(N, K)$
 5: for $t = 1, .., T$ do
 6:     Obtain batch $B$ of size $M$ traversed $k$ times from $C(N, K)$
 7:     If $k = 0$, evaluate $B$ on target $\theta'$, append target output to $B$
 8:     Compute policy and value network gradients
            $\nabla_\theta J(\theta) = \frac{1}{M} \sum_{(i,j) \in B} \frac{\nabla_\theta \pi_\theta(s_j|a_j)}{\max(\pi_{\text{target}}(s_j|a_j), \beta \pi_{\text{worker}_i}(s_j|a_j))} \hat{A}_{V\text{-GAE}} - \eta \nabla_\theta \text{KL}\left(\pi_{\text{target}}(\cdot), \pi_\theta(\cdot)\right)$
            $\nabla_w L(w) = \frac{1}{M} \sum_j (V_w(s_j) - \hat{V}_{V\text{-GAE}}(s_j)) \nabla_w V_w(s_j)$
 9:     Update policy and value network weights $\theta \leftarrow \theta + \alpha_t \nabla_\theta J(\theta)$,$w \leftarrow w - \beta_t \nabla_w L(w)$
10:     If $k = K$, discard batch $B$ from $C(N, K)$
11:     If $t \equiv 0 \pmod{t_{\text{target}}}$, update target network $(\theta', w') \leftarrow (\theta, w)$
12:     If $t \equiv 0 \pmod{t_{\text{frequency}}}$, broadcast weights to workers
13: end for
```

Worker-i

```
Input: Worker sample batch size $S$
 1: repeat
 2:     $B_i = \emptyset$
 3:     for $t = 1, ..., S$ do
 4:         Store $(s_t, a_t, r_t, s_{t+1})$ ran by $\theta_i$ in batch $B_i$
 5:     end for
 6:     Send $B_i$ to $C(N, K)$
 7:     If broadcasted weights exist, set $\theta_i \leftarrow \theta$
 8: until learner finishes
```

3 IMPACT Algorithm

Like IMPALA, IMPACT separates sampling workers from learner workers. Algorithm 1 and Figure 1c describe the main training loop and architecture of IMPACT. In the beginning, each worker copies weights from the master network. Then, each worker uses their own policy to collect trajectories and sends the data (s_t, a_t, r_t) to the circular buffer. Simultaneously, workers also asynchronously pull policy weights from the master learner. In the meantime, the target network occasionally syncs with the master learner every t_{target} iterations. The master learner then repeatedly draws experience from the circular buffer. Each sample is weighted by the importance ratio of $\frac{\pi_\theta}{\pi_{\text{worker}_i}}$ as well as clipped with target network ratio $\frac{\pi_{\text{worker}_i}}{\pi_{\text{target}}}$. The target network is used to provide a stable trust region (Figure

	PPO	**Asynchronous PPO**		
Invariants	$\pi_{\text{worker}} == \pi_{\text{master}}$	Async sampling means π_{worker} is out of sync with π_{master}		
Likelihood ratio	$\pi_\theta/\pi_{\text{worker}}$	$\pi_\theta/\pi_{\text{worker}}$	$\pi_\theta/\pi_{\text{master}}$	$\mathbf{min(\pi_\theta/\pi_{worker}, \rho\pi_\theta/\pi_{target})}$
Effectiveness	In synchronous PPO, all rollouts are fully on-policy, hence π_{worker} is the same as π_{master}.	Since π_{worker} may differ per worker, using this ratio results in trust region conflicts across multiple batches.	Since π_{master} is updated after each batch from the worker, only a single SGD step can be taken per batch.	The IMPACT objective allows for multiple SGD steps per async batch and has a stable trust region.

Figure 2: In asynchronous PPO, there are multiple candidate policies from which the trust region can be defined: (1) π_{worker_i}, the policy of the worker process that produced the batch of experiences, (2) π_{master}, the current policy of the learner process, and (3) π_{target}, the policy of a target network. Introducing the target network allows for both a stable trust region and multiple SGD steps per batch of experience collected asynchronously from workers, improving sample efficiency. Since workers can generate experiences asynchronously from their copy of the master policy, this also allows for good real-time efficiency.

2), allowing multiple steps per batch (i.e., like PPO) even in the asynchronous setting (i.e., with the IMPALA architecture). In the next section, we describe the design of this improved objective.

3.1 Maximal Target-Worker Clipping

PPO gathers experience from previous iteration's policy $\pi_{\theta_{\text{old}}}$, and the current policy trains via importance sampling experience with respect to π_θ. In the asynchronous setting, worker i's policy, denoted as π_{worker_i}, generates experience for the policy network π_θ. The probability that batch B comes from worker i can be parameterized as a categorical distribution $i \sim D(\alpha_1, ..., \alpha_n)$. We include this by adding an extra expectation to the importance-sampled policy gradient objective (IS-PG) (Jie & Abbeel, 2010):

$$J_{IS}(\theta) = \mathbb{E}_{i\sim D(\alpha)}\left[\mathbb{E}_{(s_t,a_t)\sim\pi_{\text{worker}_i}}\left[\frac{\pi_\theta}{\pi_{\text{worker}_i}}\hat{A}_t\right]\right].$$

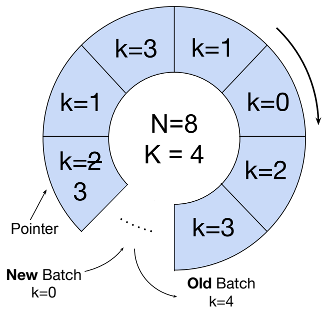
Since each worker contains a different policy, our agent introduces a target network for stability (Figure 2). Off-policy agents such as DDPG and DQN update target networks with a moving average. For IMPACT, we periodically update the target network with the master network. However, training with importance weighted ratio $\frac{\pi_\theta}{\pi_{\text{target}}}$ can lead to numerical instability, as shown in Figure 3. To prevent this, we clip the importance sampling ratio from worker policy to target policy:

$$\begin{aligned} J_{AIS}(\theta) &= \mathbb{E}_{i\sim D(\alpha)}\left[\mathbb{E}_{(s_t,a_t)\sim\pi_{\text{worker}_i}}\left[\min(\frac{\pi_{\text{worker}_i}}{\pi_{\text{target}}}, \rho)\frac{\pi_\theta}{\pi_{\text{worker}_i}}\hat{A}_t\right]\right] \\ &= \mathbb{E}_{i\sim D(\alpha)}\left[\mathbb{E}_{(s_t,a_t)\sim\pi_{\text{worker}_i}}\left[\frac{\pi_\theta}{\max(\pi_{\text{target}}, \beta\pi_{\text{worker}_i})}\hat{A}_t\right]\right], \end{aligned}$$

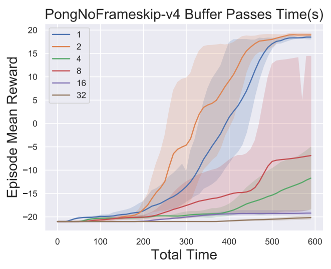
where $\beta = \frac{1}{\rho}$. In our experiments, we set ρ as a hyperparameter with $\rho \geq 1$ and $\beta \leq 1$.

For intuition, during periods of training that learner network's output changes drastically, worker i's policy, π_{worker_i}, samples data outside of the target policy, π_{target}, leading to large likelihood ratios, $\frac{\pi_{\text{worker}_i}}{\pi_{\text{target}}}$. The clipping $\min(\frac{\pi_{\text{worker}_i}}{\pi_{\text{target}}}, \rho)$ will pull back the large term back to ρ. Figure 7 provides intuition behind our target clipping objective. We show that our target network clipping is a lower bound for the IS-PG objective.

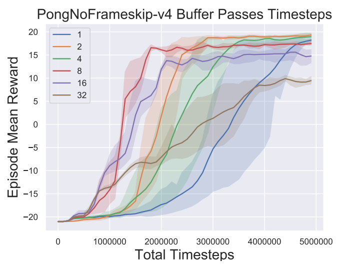
For $\rho > 1$, the clipped target ratio is larger and serves to augment advantage estimator $\hat{A}_t$. This incentivizes the agent toward good actions while avoiding bad actions. Thus, higher values of ρ encourages the agent to learn faster at the cost of instability.



(a) Circular Buffer.



(b) Sample Efficiency vs. K



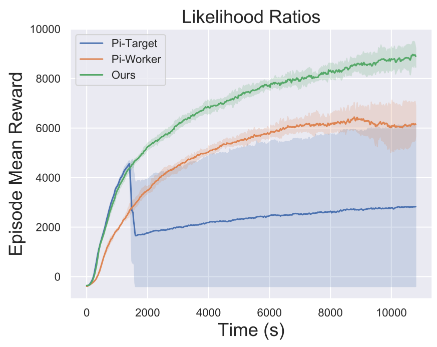
(c) Wall Clock-time vs. K

Figure 4: **Left**: The Circular Buffer in a nutshell: N and K correspond to buffer size and max times a batch can be traversed. Old batches are replaced by worker-generated batches. **Right**: IMPACT can achieve greater timestep as well as time efficiency by manipulating K. $K = 2$ outperforms all other settings in time and is more sample efficient than $K = 1, 2, 4, 16, 32$.

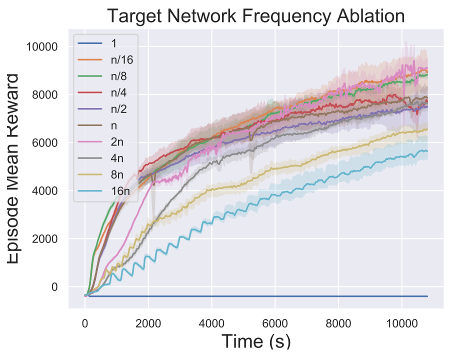
We use GAE-λ with V-trace (Han & Sung, 2019). The V-trace GAE-λ modifies the advantage function by adding clipped importance sampling terms to the summation of TD errors:

$$\hat{A}_{V\text{-GAE}} = \sum_{i=t}^{t+n-1} (\lambda\gamma)^{i-t} \left(\prod_{j=t}^{i-1} c_j \right) \delta_i,$$

where $c_i = \frac{\pi_{\text{target}}(a_i|s_i)}{\pi_{\text{worker}}(a_i|s_i)}$ and δ_i is the importance sampled 1-step TD error.



(a) Ratio ablation study.



(b) Target update frequency study.

Figure 3: Training curves of the ablation study on control benchmarks. In (a), the IMPACT objective outperforms other possible ratio choices for the surrogate loss: $R_1 = \frac{\pi_\theta}{\pi_{\text{target}}}$, $R_2 = \frac{\pi_\theta}{\pi_{\text{worker}_i}}$, $R_3 = \frac{\pi_\theta}{\max(\pi_{\text{target}}, \beta\pi_{\text{worker}_i})}$. In (b), we show the target network update frequency is robust to a range of choices. We try target network update frequency t_{target} equal to the multiple (ranging from 1/16 and 16) of $n = N \cdot K$, the product of the size of circular buffer and the replay times for each batch in the buffer.

3.2 CIRCULAR BUFFER

IMPACT uses a circular buffer (Figure 4) to emulate the mini-batch SGD used by standard PPO. The circular buffer stores N batches that can be traversed at max K times. Upon being traversed K times, a batch is discarded and replaced by a new worker batch.

For motivation, the circular buffer and the target network are analogous to mini-batching from π_{old} experience in PPO. When target network's update frequency $f = NK$, the circular buffer is equivalent to distributed PPO's training batch when the learner samples N minibatches for K SGD iterations.

This is in contrast to standard replay buffers, such as in ACER and APE-X, where transitions (s_t, a_t, r_t, s_{t+1}) are either uniformly sampled or sampled based on priority, and, when the buffer is full, the oldest transitions are discarded (Wang et al., 2016; Horgan et al., 2018).

Figure 4 illustrates an empirical example where tuning K can increase training sample efficiency and decrease training wall-clock time.

4 Evaluation

In our evaluation we seek to answer the following questions:

1. How does our target-clipping objective affect the performance of the agents compared to prior work? (Section 4.1)
2. How does the IMPACT circular buffer affect sample efficiency and training wall-clock time? (Section 4.2)
3. How does IMPACT compare to PPO and IMPALA baselines in terms of sample and real-time performance? (Section 4.3)

4.1 Target Clipping Performance

We investigate the performance of the clipped-target objective relative to prior work, which includes PPO and IS-PG based objectives. Specifically, we consider the following ratios below:

$$R_1 = \frac{\pi_\theta}{\pi_{\text{target}}} \quad R_2 = \frac{\pi_\theta}{\pi_{\text{worker}_i}} \quad R_3 = \frac{\pi_\theta}{\max(\pi_{\text{target}}, \beta\pi_{\text{worker}_i})}$$

For all three experiments, we truncate all three ratios with PPO's clipping function: $c(r) = \text{clip}(r, 1-\epsilon, 1+\epsilon)$ and train in an asynchronous setting. Figure 4(a) reveals two important takeaways: first, R_1 suffers from sudden drops in performance midway through training. Next, R_2 trains stably but does not achieve good performance.

We theorize that R_1 fails due to the target and worker network mismatch. During periods of training where the master learner undergoes drastic changes, worker's action outputs vastly differ from the learner's outputs, resulting in small action probabilities. This creates large ratios in training and destabilizes training. We hypothesize that R_2 fails due to different workers pushing and pulling the learner in multiple directions. The learner moves forward the most recent worker's suggestions without developing a proper trust region, resulting in many worker's suggestions conflicting with each other.

Our loss function, R_3 shows that clipping is necessary and can help facilitate training. By clipping the target-worker ratio, we make sure that our ratio does not explode and destabilize training. Furthermore, we prevent workers from making mutually-destructing suggestions by having a target network provide singular guidance.

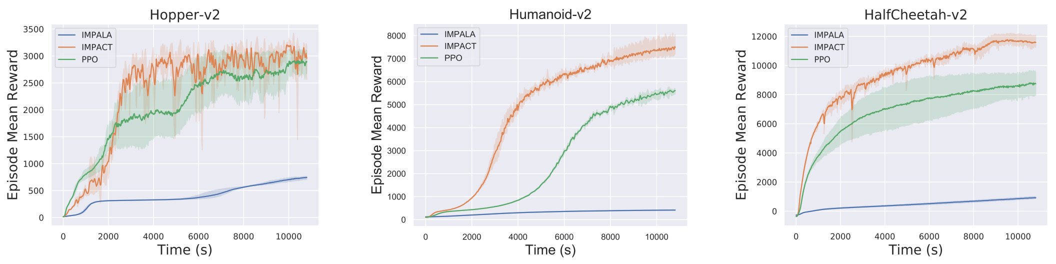
4.1.1 Target Network Update Frequency

In Section 3.2, an analogy was drawn between PPO's mini-batching mechanism and the circular buffer. Our primary benchmark for target update frequency is $n = N \cdot K$, where N is circular buffer size and K is maximum replay coefficient. This is the case when PPO is equivalent to IMPACT.

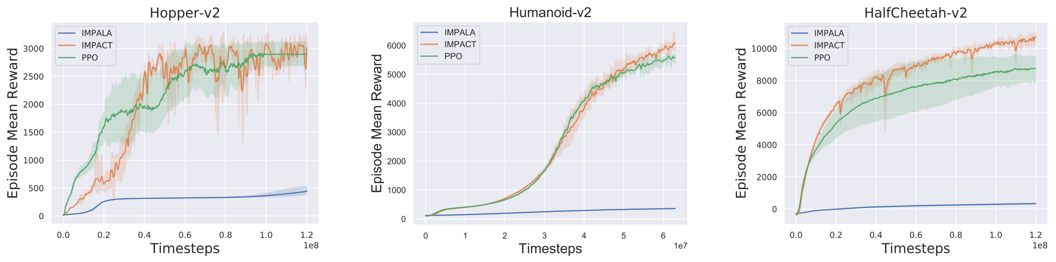
In Figure 4(b), we test the frequency of updates with varying orders of magnitudes of n. In general, we find that agent performance is robust to vastly differing frequencies. However, when $n = 1 \sim 4$, the agent does not learn. Based on empirical results, we theorize that the agent is able to train as long as a stable trust region can be formed. On the other hand, if update frequency is too low, the agent is stranded for many iterations in the same trust region, which impairs learning speed.

4.2 Time and Sample Efficiency with Circular Buffer

Counter to intuition, the tradeoff between time and sample efficiency when K increases is not necessarily true. In Figure 4b and 4c, we show that IMPACT realizes greater gains by striking the balance between high sample throughput and sample efficiency. When $k = 2$, IMPACT performs the best in both time and sample efficiency. Our results reveal that wall-clock time and sample efficiency can be optimized based on tuning values of K in the circular buffer.



(a) Time



(b) Timesteps

Figure 5: IMPACT outperforms baselines in both sample and time efficiency in the Continuous Control Domain: Hopper, Humanoid, HalfCheetah.

4.3 Comparison with Baselines

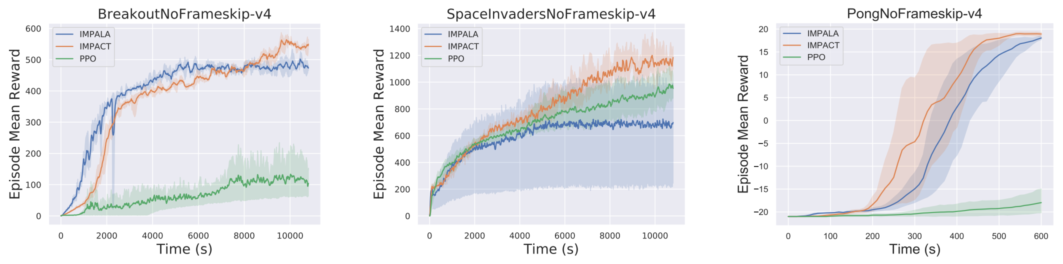
We investigate how IMPACT strikes the balance between wall clock-time and sample efficiency compared with PPO and IMPALA across six different continuous control and discrete action tasks.

We tested our agent on three continuous environments (Figure 5): HalfCheetah-v2, Hopper-v2, and Humanoid-v2 on 16 CPUs and 1 Titan XP. The policy networks were FC networks with two hidden layers of 256 units and nonlinear activation tanh. For the value network, we used a similar separate fully connected network without weight sharing. For fairness, same network hyperparameters were used across PPO, IMPALA, and IMPACT.

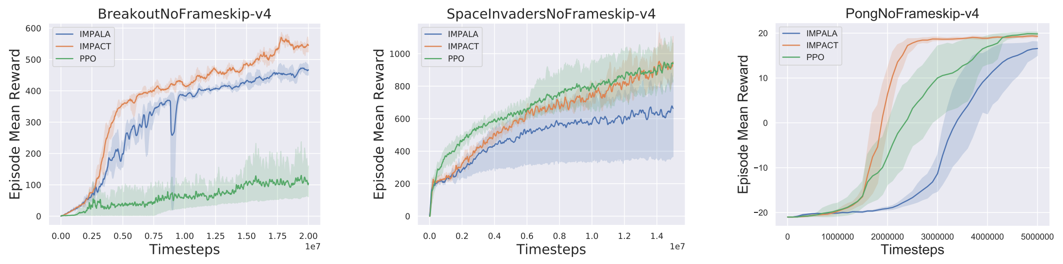
For the discrete environments (Figure 6), we ran our experiments on 32 CPUs and 1 Titan XP. Our policy network consists of three 4x4 and one 11x11 convolutional layer, with ReLU as our nonlinear activation. Our critic network shares weights with the policy network. The input of the network is a stack of four 42x42 down-sampled images of the Atari environment. Our hyper-parameters for continuous and discrete environments are listed in the Appendix table 3 and 6 respectively.

Figure 5 and 6 show the total average return of evaluation rollouts for IMPACT, IMPALA and PPO. We train each algorithm with three different random seeds on each environment with the total time of three hours, except 10 minutes on Pong environments. According to the experiments, our algorithm is able to achieve greater timestep efficiency than synchronous distributed PPO, while training faster than IMPALA.

Our results show that continuous control tasks for IMPACT can be sensitive to the choice of N and K for the circular buffer, and $N = 8$ and $K = 20$ seemed to be a robust choice for a variety of tasks. Although the high K reduces the sample throughput of the algorithm, the increased sample efficiency results in an overall reduction in training wall-clock time and improved performance. According to the results from the discrete control tasks, $N = 1$ and $K = 2$ seemed to be a good choice for the circular buffer. Unlike continuous control environments, sampling new data instead of replaying old data helps the agents learn faster.



(a) Time



(b) Timesteps

Figure 6: IMPACT outperforms PPO and IMPALA in both real-time and sample efficiency in the Discrete Control Domain: Breakout, SpaceInvaders, and Pong.

5 Related Work

Distributed RL architectures are often used to accelerate training. Gorila (Nair et al., 2015) and A3C (Mnih et al., 2016) use workers to compute gradients to be sent to the learner. A2C (Mnih et al., 2016) and IMPALA (Espeholt et al., 2018) send experience tuples to the learner. Distributed replay buffers, introduced in ACER (Wang et al., 2016) and Ape-X (Horgan et al., 2018), collect worker-collected experience and define an overarching heuristic for learner batch selection. IMPACT is the first to fully incorporate the sample-efficiency benefits of PPO in an asynchronous setting.

Surreal PPO (Fan et al., 2018) also studies training with PPO in the asynchronous setting, but do not consider adaptation of the surrogate objective nor IS-correction. Their use of a target network for broadcasting weights to workers is also entirely different from IMPACT's. Consequently, IMPACT is able to achieve better results in both real-time and sample efficiency.

Off-policy methods, including DDPG(Lillicrap et al., 2015) and QProp, utilize target networks to stabilize learning the Q function (Lillicrap et al., 2015; Gu et al., 2016). This use of a target network is related but different from IMPACT, which uses the network to define a stable trust region for the PPO surrogate objective.

6 Conclusion

In conclusion, we introduce IMPACT, which extends PPO with a stabilized surrogate objective for asynchronous optimization, enabling greater real-time performance without sacrificing timestep efficiency. We show the importance of the IMPACT objective to stable training, and show it can outperform tuned PPO and IMPALA baselines in both real-time and timestep metrics.

References

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, pp. 767–782, 2018.

Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

Seungyul Han and Youngchul Sung. Amber: Adaptive multi-batch experience replay for continuous action control. *arXiv preprint arXiv:1710.04423*, 2017.

Seungyul Han and Youngchul Sung. Dimension-wise importance sampling weight clipping for sample-efficient reinforcement learning. *arXiv preprint arXiv:1905.02363*, 2019.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

Tang Jie and Pieter Abbeel. On a connection between importance sampling and the likelihood ratio policy gradient. pp. 1000–1008, 2010.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

A Experiment hyper-parameters for continuous environments

	Hopper	Humanoid	Half-Cheetah
clip param	0.4	0.3	0.4
entropy coeff	0.0	0.0	0.01
grad clip	0.5	1.0	0.5
lambda	1.0	0.95	0.95
learning rate	0.00002	0.0001	0.0008
minibatch buffer size	5	16	16
num sgd iter	20	20	32
sample batch size	4096	1024	512
train batch size	32768	32768	4096
kl coeff	1.0	1.0	1.0
kl target	0.01	0.04	0.04

Table 1: Hyper-parameters for continuous control tasks (IMPACT)

	Hopper	Humanoid	Half-Cheetah
entropy coeff	0.0	0.0	0.0
grad clip	0.5	0.5	0.5
gamma	0.995	0.995	0.995
learning rate	0.00002	0.00001	0.000013
sample batch size	4096	4096	4096
train batch size	32768	32768	32768
value function loss coeff	1.0	0.5	0.5

Table 2: Hyper-parameters for continuous control tasks (IMPALA)

	Hopper	Humanoid	Half-Cheetah
clip params	0.3	0.2	0.3
entropy coeff	0.00	0.00	0.00
lambda	1.0	1.0	1.0
learning rate	0.0001	0.00005	0.0003
num sgd iter	20	20	32
sgd minibatch size	32768	32768	4096
train batch size	160000	320000	65536
kl coeff	1.0	1.0	1.0
kl target	0.01	0.04	0.04

Table 3: Hyper-parameters for continuous control tasks (PPO)

	Breakout	Spaceinvader	Pong
clip param	0.2	0.2	0.3
entropy coeff	0.01	0.01	0.01
grad clip	2.5	2.5	10
lambda	0.995	0.995	1.0
learning rate	0.0005	0.0005	0.0005
num sgd iter	1	2	2
sample batch size	50	50	50
train batch size	500	800	500
value function loss coeff	0.25	1.0	1.0
kl coeff	0.0	0.0	1.0
kl target	0.01	0.01	0.01

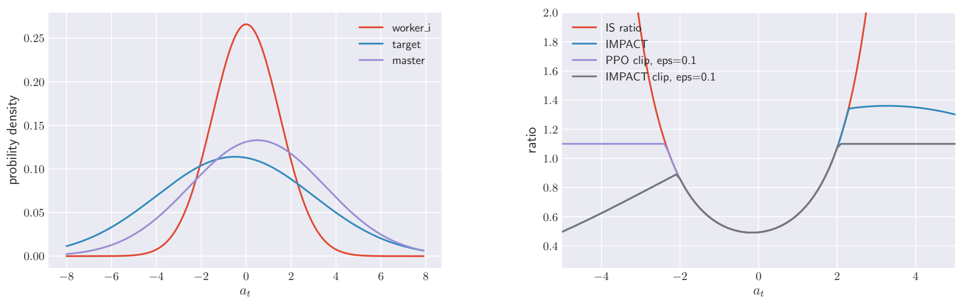
Table 4: Hyper-parameters for discrete control tasks (IMPACT)

	Breakout	Spaceinvader	Pong
entropy coeff	0.01	0.01	0.01
grad clip	40.0	40.0	40.0
learning rate	0.0005	0.0005	0.0005
sample batch size	50	50	50
train batch size	500	500	750
value function loss coeff	0.5	0.5	0.5

Table 5: Hyper-parameters for discrete control tasks (IMPALA)

	Breakout	Spaceinvader	Pong
clip param	0.1	0.1	0.1
entropy coeff	0.01	0.01	0.01
lambda	0.95	0.95	0.95
learning rate	0.00005	0.00005	0.00005
num sgd iter	10	10	10
sample batch size	100	100	20
train batch size	5000	5000	5000
kl coeff	0.5	0.5	0.5
kl target	0.01	0.01	0.01

Table 6: Hyper-parameters for discrete control tasks(PPO)



Action Distributions

Likelihood Ratios w.r.t Different Objectives

Figure 7: Likelihood ratio $r_t(\theta)$ for different objective functions, including PPO's. We assume a diagonal Gaussian policy for our policy. **Left**: Corresponding one dimensional action distributions for Worker i, Target, and Master Learner; **Right**: Ratio values graphed as a function of possible action values. IMPACT with PPO clipping is a lower bound of PPO.

B The intuition of the objective

In the Figure 7, the formula we make the plot is as follows. Without the loss of the generality, we can assume the advantage function is one. Then, the following ratio just represents the objective function.

- IS ratio: $\frac{\pi_\theta}{\pi_{\text{worker}_i}}$
- IMPACTtarget: $\min\left(\frac{\pi_{\text{worker}_i}}{\pi_{\text{target}}}, \rho\right) \frac{\pi_\theta}{\pi_{\text{worker}_i}}$
- PPO clip with epsilon: $\min\left(\frac{\pi_\theta}{\pi_{\text{worker}_i}}, \text{clip}(\frac{\pi_\theta}{\pi_{\text{worker}_i}}, 1-\epsilon, 1+\epsilon)\right)$
- IMPACT target clip: $\min\left(\min\left(\frac{\pi_{\text{worker}_i}}{\pi_{\text{target}}}, \rho\right) \frac{\pi_\theta}{\pi_{\text{worker}_i}}, \text{clip}\left(\min\left(\frac{\pi_{\text{worker}_i}}{\pi_{\text{target}}}, \rho\right) \frac{\pi_\theta}{\pi_{\text{worker}_i}}, 1-\epsilon, 1+\epsilon\right)\right)$

We can see from the Figure 7, the first ratio is bigger where π_{worker_i} has little probability. Among all the curves, we can find the last one is the lower bounds for all the others. This gives the intuition that using the IMPACT target clip ratio as the objective can be more stable because one gradient step ahead won't change the master policy too much.