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, Importance Weighted Asynchronous Architectures with Clipped
Target 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).
In Figure 1a, distributed PPO agents implement a synchronous data-gathering scheme. Before data collection, workers are updated to \( \pi_{\theta_{old}} \) 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 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 high variance and large update-step sizes, oftentimes leading to sudden drops in performance.

Let \( \pi_{\theta} \) denote a stochastic policy mapping that returns an action distribution given state \( s \). Acting out with \( \pi_{\theta} \) is equivalent to sampling a trajectory \( T \sim \mathbb{P}(\tau|\pi) \), where \( \tau := (s_0, a_0, ..., a_{T-1}, s_T, a_T) \). We can compactly define the trajectory distribution \( \mathbb{P}(\tau|\pi) \) with state and state-action marginals \( p_S(s_t) \) and \( p_A(s_t,a_t) \). The goal for reinforcement learning aims to maximize the following objective:

\[
J(\pi) = E_{(s_t,a_t)}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]
\]

When \( \theta \) parameterizes \( \pi_{\theta} \), performing a gradient w.r.t \( \theta \) is equivalent to the Policy Gradient Theorem (Sutton & Barto, 2018):

\[
\nabla_{\theta} J(\theta) = \mathbb{E}_{(s_t,a_t)}[\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) \hat{A}_{\pi_{\theta}}(s_t, a_t)],
\]

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 \( \pi_{\theta} \) from target \( \pi_{\theta_{old}} \) via the following objective function

\[
L(\theta) = \mathbb{E}_{(s_t,a_t)}\left[ \min\left( r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) \hat{A}_t \right) \right],
\]

where \( r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \) and \( \epsilon \) is the clipping hyperparameter. In addition, many PPO implementations use GAE-\( \lambda \) 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 \( \pi_{\theta_{old}} \) is too far from \( \pi_{\theta} \). (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 \( \pi_{\theta_{old}} \) 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.
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:

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

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

**Algorithm 1 IMPACT**

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

**Worker-i**

**Input:** Worker sample batch size \( S \)
1: \text{repeat} \\
2: \( B_i = \emptyset \) \\
3: \text{for } t = 1, ..., S \text{ do} \\
4: \text{Store } (s_i, a_t, r_t, s_{t+1}) \text{ ran by } \theta_i \text{ in batch } B_i \\
5: \text{end for} \\
6: \text{Send } B_i \text{ to } C(N, K) \\
7: \text{If broadcasted weights exist, set } \theta_i \leftarrow \theta \\
8: \text{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 \( \frac{\pi_\theta}{\pi_{target}} \) as well as clipped with target network ratio \( \frac{\pi_{worker}}{\pi_{target}} \). The target network is used to provide a stable trust region (Figure...
Figure 2: In asynchronous PPO, there are multiple candidate policies from which the trust region can be defined: (1) $\pi_{worker}$, the policy of the worker process that produced the batch of experiences, (2) $\pi_{master}$, the current policy of the learner process, and (3) $\pi_{target}$, 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.

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 $\pi_{\theta}$. In the asynchronous setting, worker $i$’s policy, denoted as $\pi_{worker}$, generates experience for the policy network $\pi_{\theta}$. The probability that batch $B$ comes from worker $i$ can be parameterized as a categorical distribution $i \sim D(\alpha_1, \ldots, \alpha_n)$. 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 $\frac{\pi_{\theta}}{\pi_{target}}$ 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:

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

where $\beta = \frac{1}{\rho}$. In our experiments, we set $\rho$ 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, $\pi_{\text{worker}_i}$, samples data outside of the target policy, $\pi_{\text{target}}$, 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 $\rho$. 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 $\rho$ encourages the agent to learn faster at the cost of instability.
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 $\delta_i$ is the importance sampled 1-step TD error.

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 $\pi_{\text{old}}$ 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:

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

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.

![Graphs showing time and sample efficiency for different environments](image)

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.

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


### A  EXPERIMENT HYPER-PARAMETERS FOR CONTINUOUS ENVIRONMENTS

<table>
<thead>
<tr>
<th>clip param</th>
<th>Hopper</th>
<th>Humanoid</th>
<th>Half-Cheetah</th>
</tr>
</thead>
<tbody>
<tr>
<td>entropy coeff</td>
<td>0.4</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>grad clip</td>
<td>0.0</td>
<td>0.0</td>
<td>0.01</td>
</tr>
<tr>
<td>lambda</td>
<td>0.5</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>learning rate</td>
<td>1.0</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>minibatch buffer size</td>
<td>0.00002</td>
<td>0.0001</td>
<td>0.0008</td>
</tr>
<tr>
<td>num sgd iter</td>
<td>5</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>sample batch size</td>
<td>4096</td>
<td>1024</td>
<td>512</td>
</tr>
<tr>
<td>train batch size</td>
<td>32768</td>
<td>32768</td>
<td>4096</td>
</tr>
<tr>
<td>kl coeff</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>kl target</td>
<td>0.01</td>
<td>0.04</td>
<td>0.04</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>entropy coeff</th>
<th>Hopper</th>
<th>Humanoid</th>
<th>Half-Cheetah</th>
</tr>
</thead>
<tbody>
<tr>
<td>grad clip</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>gamma</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>learning rate</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>sample batch size</td>
<td>0.00002</td>
<td>0.00001</td>
<td>0.000013</td>
</tr>
<tr>
<td>train batch size</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
</tr>
<tr>
<td>value function loss coeff</td>
<td>1.0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>clip params</th>
<th>Hopper</th>
<th>Humanoid</th>
<th>Half-Cheetah</th>
</tr>
</thead>
<tbody>
<tr>
<td>entropy coeff</td>
<td>0.3</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>lambda</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>learning rate</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>num sgd iter</td>
<td>20</td>
<td>20</td>
<td>32</td>
</tr>
<tr>
<td>sgd minibatch size</td>
<td>32768</td>
<td>32768</td>
<td>4096</td>
</tr>
<tr>
<td>train batch size</td>
<td>160000</td>
<td>320000</td>
<td>65536</td>
</tr>
<tr>
<td>kl coeff</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>kl target</td>
<td>0.01</td>
<td>0.04</td>
<td>0.04</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>clip param</th>
<th>Breakout</th>
<th>Spaceinvader</th>
<th>Pong</th>
</tr>
</thead>
<tbody>
<tr>
<td>entropy coeff</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>grad clip</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>lambda</td>
<td>2.5</td>
<td>2.5</td>
<td>10</td>
</tr>
<tr>
<td>learning rate</td>
<td>0.995</td>
<td>0.995</td>
<td>1.0</td>
</tr>
<tr>
<td>num sgd iter</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>sample batch size</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>train batch size</td>
<td>500</td>
<td>800</td>
<td>500</td>
</tr>
<tr>
<td>value function loss coeff</td>
<td>0.25</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>kl coeff</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>kl target</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Table 4: Hyper-parameters for discrete control tasks (IMPACT)
<table>
<thead>
<tr>
<th>Hyper-parameters for discrete control tasks (IMPALA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakout</td>
</tr>
<tr>
<td>entropy coeff</td>
</tr>
<tr>
<td>grad clip</td>
</tr>
<tr>
<td>learning rate</td>
</tr>
<tr>
<td>sample batch size</td>
</tr>
<tr>
<td>train batch size</td>
</tr>
<tr>
<td>value function loss coeff</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hyper-parameters for discrete control tasks(PPO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakout</td>
</tr>
<tr>
<td>clip param</td>
</tr>
<tr>
<td>entropy coeff</td>
</tr>
<tr>
<td>lambda</td>
</tr>
<tr>
<td>learning rate</td>
</tr>
<tr>
<td>num sgd iter</td>
</tr>
<tr>
<td>sample batch size</td>
</tr>
<tr>
<td>train batch size</td>
</tr>
<tr>
<td>kl coeff</td>
</tr>
<tr>
<td>kl target</td>
</tr>
</tbody>
</table>

**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.

**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**: $\frac{\pi_{\theta}}{\pi_{\text{worker}_i}}$
- **IMPACT Target**: $\min\left(\frac{\pi_{\text{worker}_i}}{\pi_{\text{target}}}, \rho \right)$ $\frac{\pi_{\theta}}{\pi_{\text{worker}_i}}$
- **PPO clip with epsilon**: $\min\left(\frac{\pi_{\theta}}{\pi_{\text{worker}_i}}, \text{clip} \left(\frac{\pi_{\theta}}{\pi_{\text{worker}_i}}, 1 - \epsilon, 1 + \epsilon \right)\right)$
- **IMPACT target clip**: $\min\left(\min\left(\frac{\pi_{\theta}}{\pi_{\text{worker}_i}}, \rho \right), \frac{\pi_{\theta}}{\pi_{\text{worker}_i}}, \text{clip} \left(\min\left(\frac{\pi_{\theta}}{\pi_{\text{target}}}, \rho \right), 1 - \epsilon, 1 + \epsilon \right)\right)$
We can see from the Figure 7, the first ratio is bigger where $\pi_{\text{worker}}$, 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.