Sample Efficient Actor Critic with Experience Replay

Anurodh Jain

Prashanth N Bhat

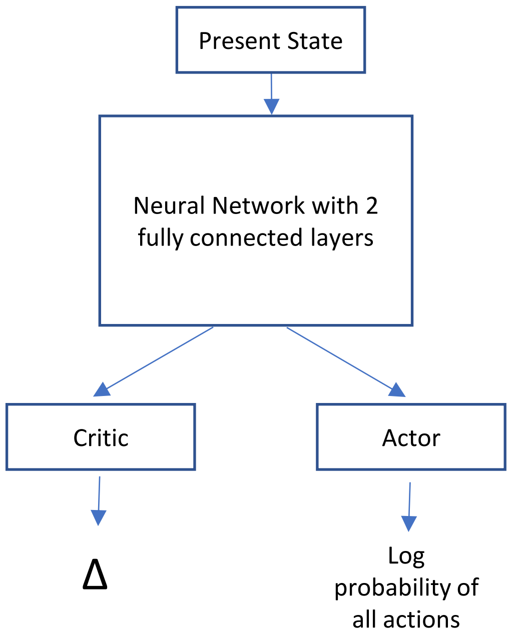
Ronak Dedhiya Dinesh

Goals of the project

- Understand the paper
- Implementation of various optimizations and their impact
  - Base Actor-Critic
  - Actor-Critic with optimizations
    - Naïve Experience Replay
    - Importance Sampling with bias correction
    - Trust Region Optimization
  - Sample Efficient Actor Critic with Experience Replay
  - Hyperparameter sensitivity
  - Conclusion

- Experiment setup
  - Environment: Cartpole-v1
  - Training episodes: 500 episodes
    - Plotted moving average of 100 episodes for rewards
  - Test episodes: 1000 episodes
    - To capture mean/std. Dev. Of rewards.

Actor critic


Present State
Neural Network with 2 fully connected layers
Critic
Actor
Δ
Log probability of all actions

```
delta = reward + self.gamma*critic_value_*(1-int(done)) - critic_value

actor_loss = -self.log_prob*delta
critic_loss = delta**2
```

Implementation:

- Implemented as a Neural Network with 2 fully connected layers
- Output of 2nd fully connected layers (32 neurons) is given as input to both actor and critic
- Actor:
  - No. Of output neurons = no. Of actions
  - Output is probability of all actions
  - Actions are chosen based on probability
- Critic:
  - No. Of output neurons = 1
  - Output is value function for each state.

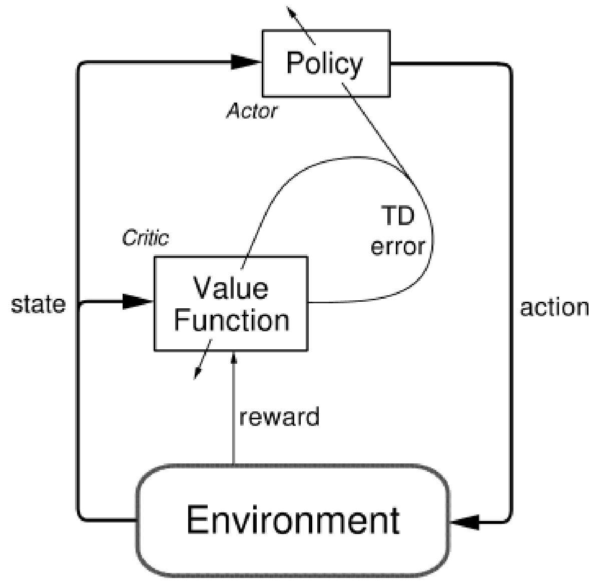
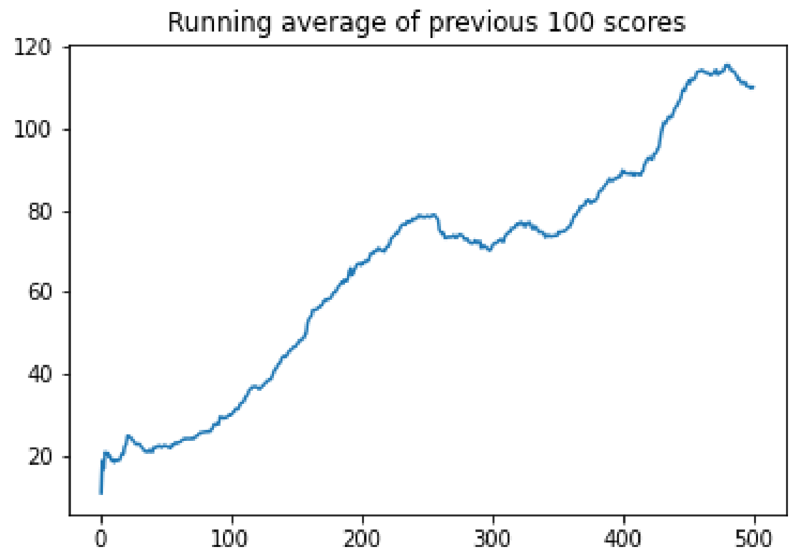


Image from Sutton & Barto: Reinforcement Learning: An Introduction http://www.incompleteideas.net/book/ebook/node66.html

Results

TRAINING



Training progress

Testing:

- Mean of 1000 episodes: 23.65
- Std. Dev. Of 1000 episodes: 13.28

Naïve Experience Replay

- On-policy asynchronous advantage actor critic (A3C) of Mnih et al. (2016), are sample inefficient
- Off-policy learning with experience replay may appear to be an obvious strategy for improving the sample efficiency of actor-critics. However, controlling the variance and stability of off-policy estimators is hard.

Naïve experience replay:

- All the trajectories (S, A, R, S') are stored in a Replay Buffer.
- In a naïve implementation of experience replay:
    - All trajectories are stored in a Replay buffer.
    - During replay:
        - Starting trajectory is chosen at random.
        - A trajectory batch starting from the start trajectory is used to train the actor-critic network.
    - The performance is not good as the average reward shows a very high variance, and hence the training oscillates and doesn't settle. It may also lead to correlated samples and stale gradients

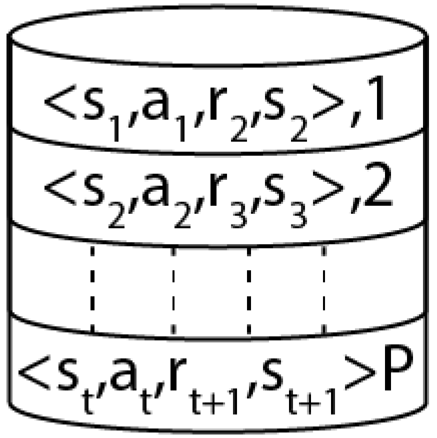
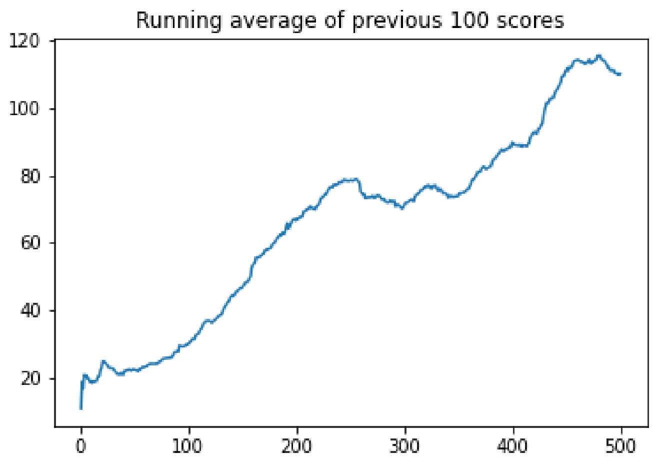
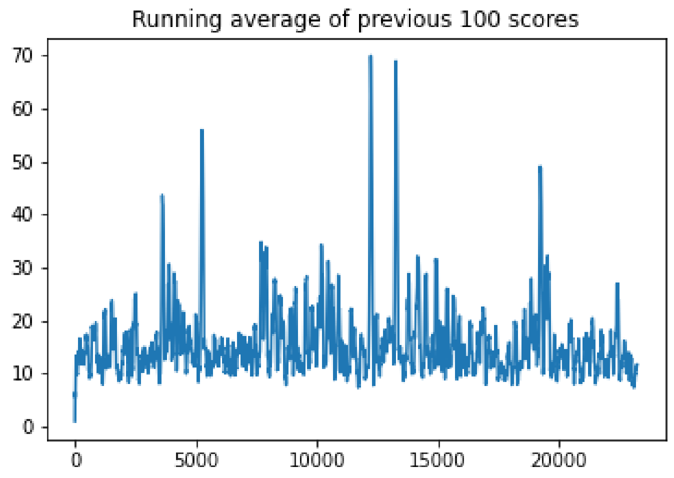


Image from: Hands on Reinforcement Learning with Python by Sudarshan Ravichandiran https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788836524

Results with naive experience replay



No Experience replay



Naïve Experience replay

- With naive experience replay, the network is not stable, and rewards are low

Importance Sampling

- Using past experience makes the problem as off-policy method which means utilizing samples from a different distribution than the one being evaluated
- Importance sampling provides the way to weigh the samples according to their importance. It is given by:

$$\widehat{g}^{\mathrm{imp}} = \left(\prod_{t=0}^{k} \rho_t\right) \sum_{t=0}^{k} \left(\sum_{i=0}^{k} \gamma^i r_{t+i}\right) \nabla_\theta \log \pi_\theta(a_t|x_t),$$

where $\rho_t = \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}$ denotes the importance weight.

Cons:

1. Suffers from high variance due to product of potentially unbounded importance weight

2. Truncating this product bounds the variance, but could suffer from significant bias

Marginal Importance weight and Retrace

- The above problem of high variance can be fixed by using marginal value function over the limiting distribution*.
- Here it depends on Q^{π} and not on Q^{μ}
- We no longer have product of importance weight, but instead only need to estimate the marginal importance weight

$$g^{\text{marg}} = \mathbb{E}_{x_t \sim \beta, a_t \sim \mu} \left[\rho_t \nabla_\theta \log \pi_\theta(a_t | x_t) Q^{\pi}(x_t, a_t) \right]$$

Retrace:

- Author estimate the Q^{π} using retrace algorithm**.
- Retrace is an off-policy, return-based algorithm which has low variance and is proven to converge (in the tabular case) to the value function of the target policy for any behavior policy. It is estimated as follows:

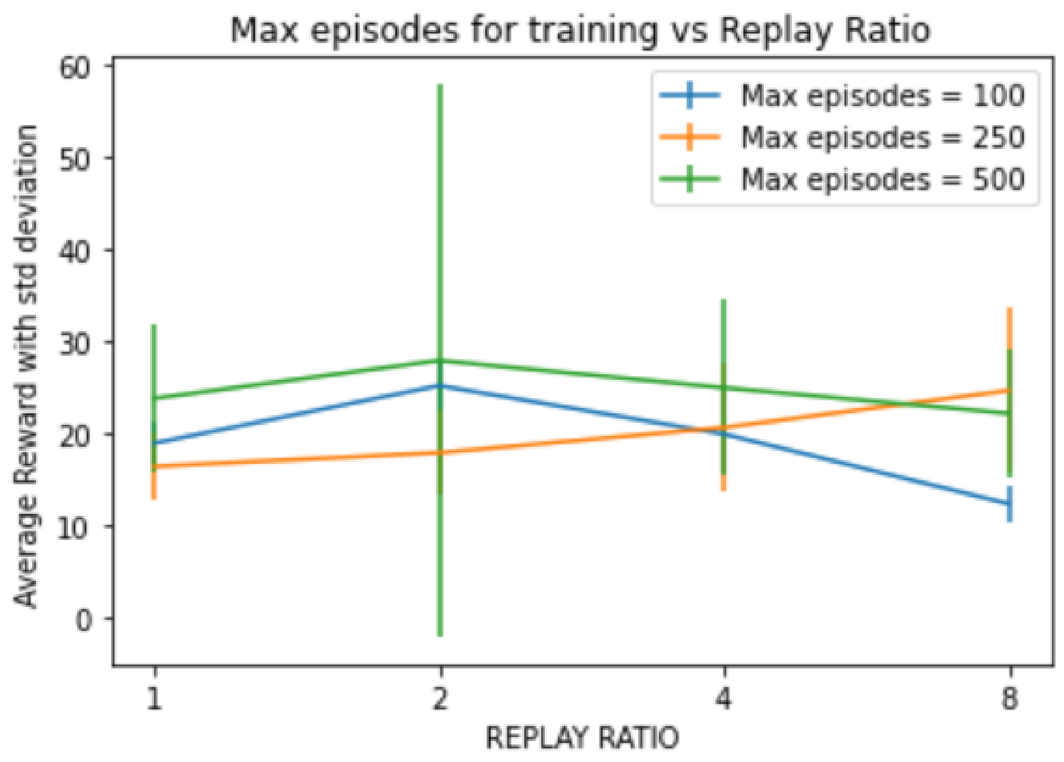
$$Q^{\text{ret}}(x_t, a_t) = r_t + \gamma \bar{\rho}_{t+1} [Q^{\text{ret}}(x_{t+1}, a_{t+1}) - Q(x_{t+1}, a_{t+1})] + \gamma V(x_{t+1})$$

*T. Degris, M. White, and R. S. Sutton. Off-policy actor-critic. In ICML, pp. 457–464, 2012.

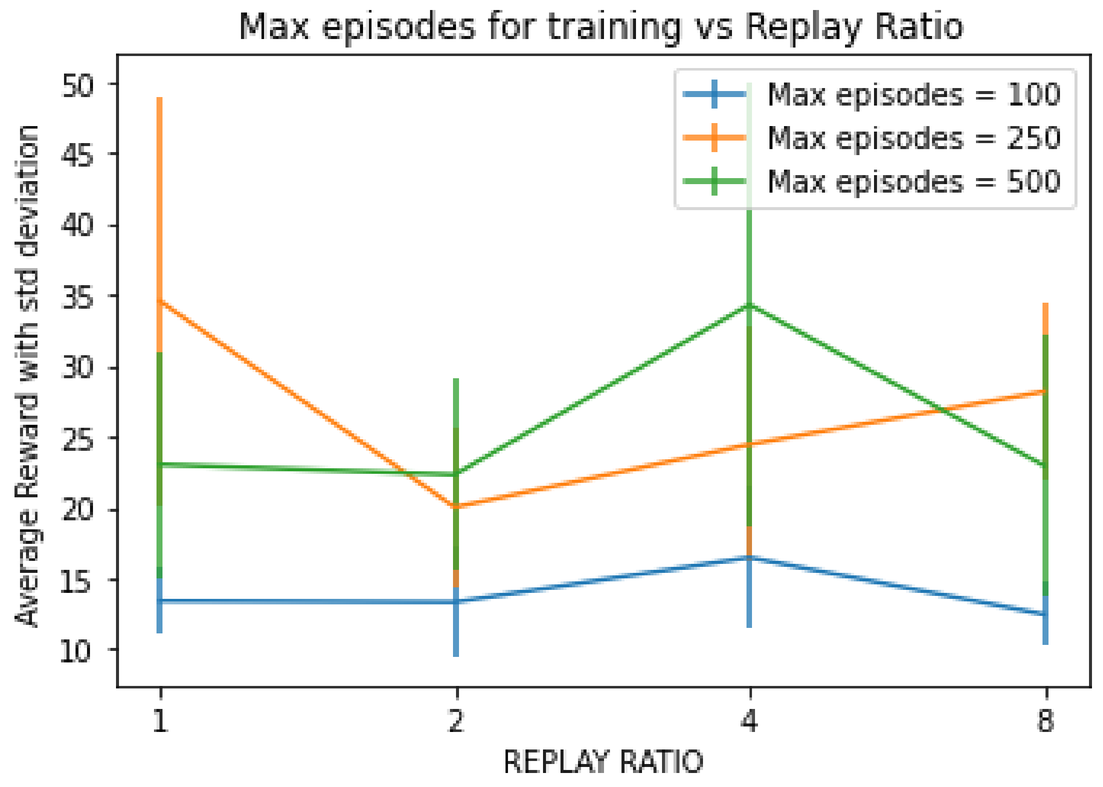
** R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and efficient off-policy reinforcement learning. arXiv preprint arXiv:1606.02647, 2016

Naïve Experience Replay vs Marginal Importance Sampling

Naïve Experience Replay



Marginal Importance sampling



- Importance sampling performs badly when trained with only 100 episodes.
- The average reward is comparable, but bit higher mean value is obtained with importance sampling. It also results in higher variance.

ACER

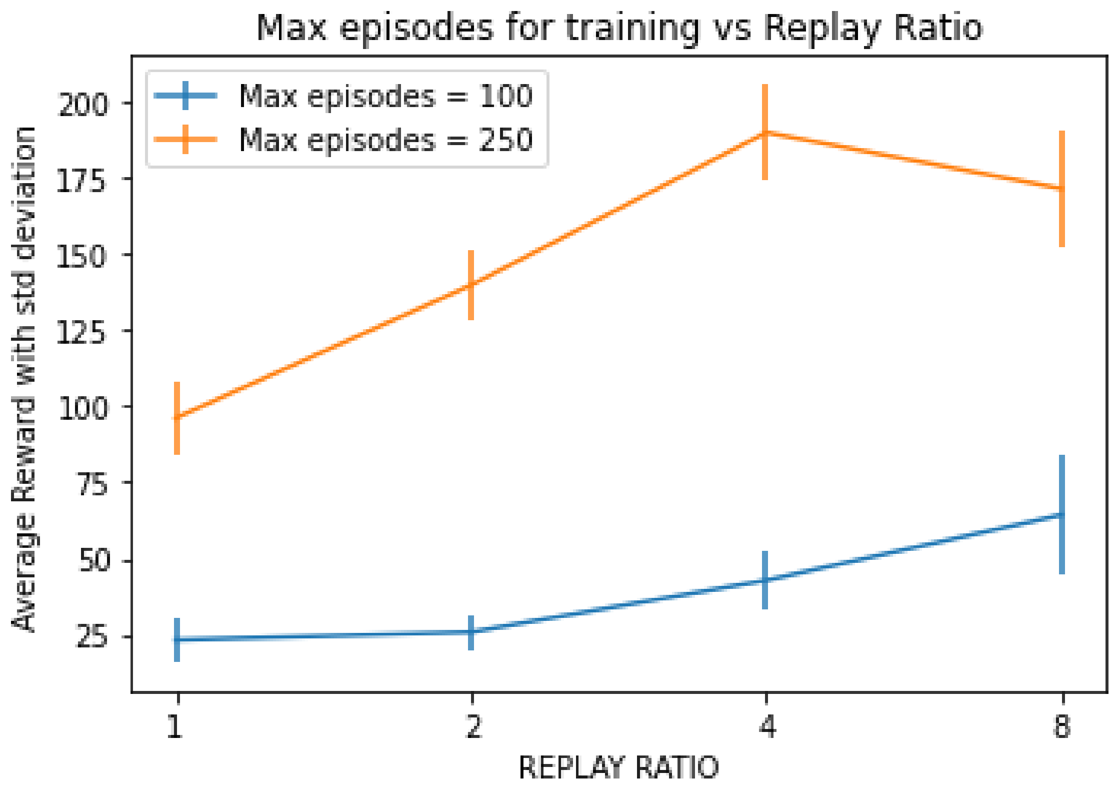
- **Bias correction:** The marginal importance weights can also lead to high variance, it is proposed to truncate the importance weights and introduce a correction term via the following decomposition:

$$
\begin{aligned}
\widehat{g}_t^{\text{acer}} \;=\;& \overbrace{\bar{\rho}_t}^{\text{Ensures bounded}} \nabla_\theta \log \pi_\theta(a_t|x_t)\Big[\overbrace{Q^{\text{ret}}(x_t,a_t)}^{\text{Retrace}} - \overbrace{V_{\theta_v}(x_t)}^{\text{Classical baseline to reduce variance}}\Big] \\
&+ \mathop{\mathbb{E}}_{a\sim\pi}\left(\underbrace{\left[\frac{\rho_t(a)-c}{\rho_t(a)}\right]_+}_{\text{Ensures unbiased}} \nabla_\theta \log \pi_\theta(a|x_t)\big[Q_{\theta_v}(x_t,a) - \overbrace{V_{\theta_v}(x_t)}^{}\big]\right).
\end{aligned}
$$

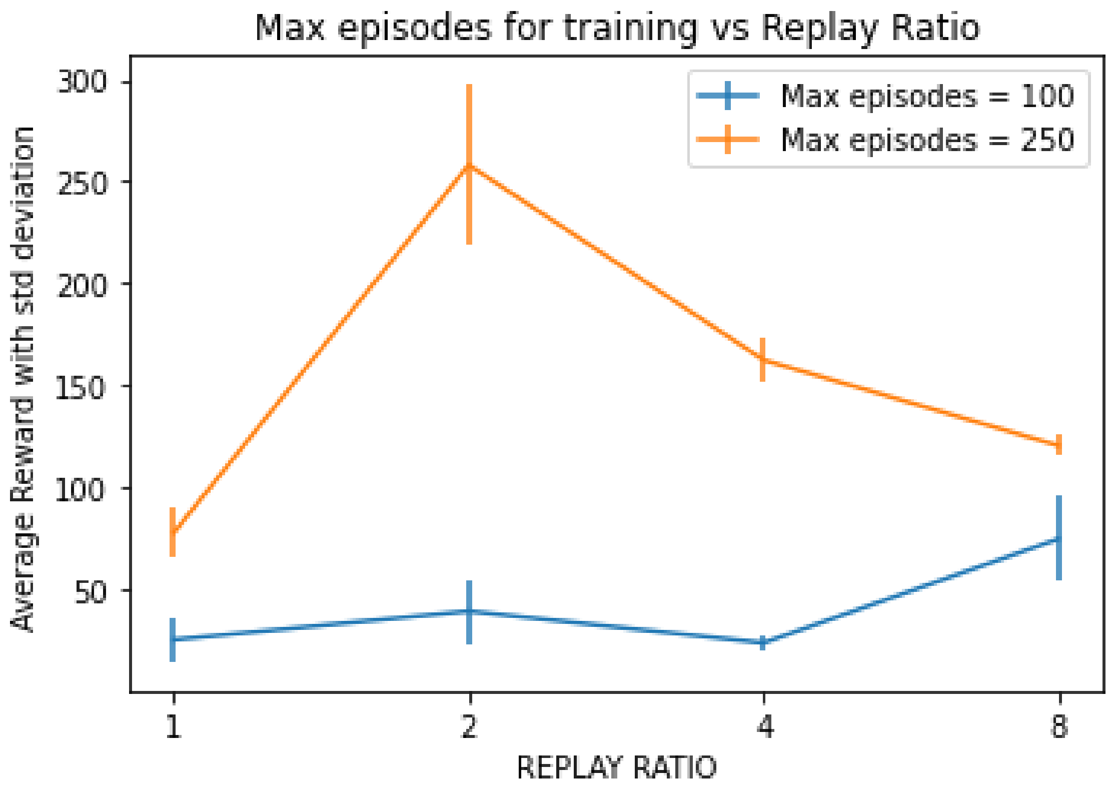
$$\text{where } \bar{\rho}_t = \min\{c, \rho_t\} \text{ with } \rho_t = \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}$$

Retrace vs ACER

ACER without Bias (Retrace)



ACER



ACER with Efficient Trust region optimization

- Simply using smaller learning rates is insufficient as they cannot guard against the occasional large updates while maintaining a desired learning speed.
- Author proposes to maintain an average policy network that represents a running average of past policies and forces the updated policy to not deviate far from this average.

Algorithm:

$$\begin{aligned} g &\leftarrow \min\left\{c, \rho_i(a_i)\right\} \nabla_{\phi_{\theta'}(x_i)} \log f(a_i|\phi_{\theta'}(x_i))(Q^{ret} - V_i) \\ &\quad + \sum_a \left[1 - \frac{c}{\rho_i(a)}\right]_+ f(a|\phi_{\theta'}(x_i)) \nabla_{\phi_{\theta'}(x_i)} \log f(a|\phi_{\theta'}(x_i))(Q_{\theta'_v}(x_i, a_i) - V_i) \\ k &\leftarrow \nabla_{\phi_{\theta'}(x_i)} D_{KL}\left[f(\cdot|\phi_{\theta_a}(x_i) \| f(\cdot|\phi_{\theta'}(x_i))\right] \end{aligned}$$

Accumulate gradients wrt θ': $d\theta' \leftarrow d\theta' + \frac{\partial \phi_{\theta'}(x_i)}{\partial \theta'}\left(g - \max\left\{0, \frac{k^T g - \delta}{\|k\|_2^2}\right\} k\right)$

Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \nabla_{\theta'_v}(Q^{ret} - Q_{\theta'_v}(x_i, a))^2$

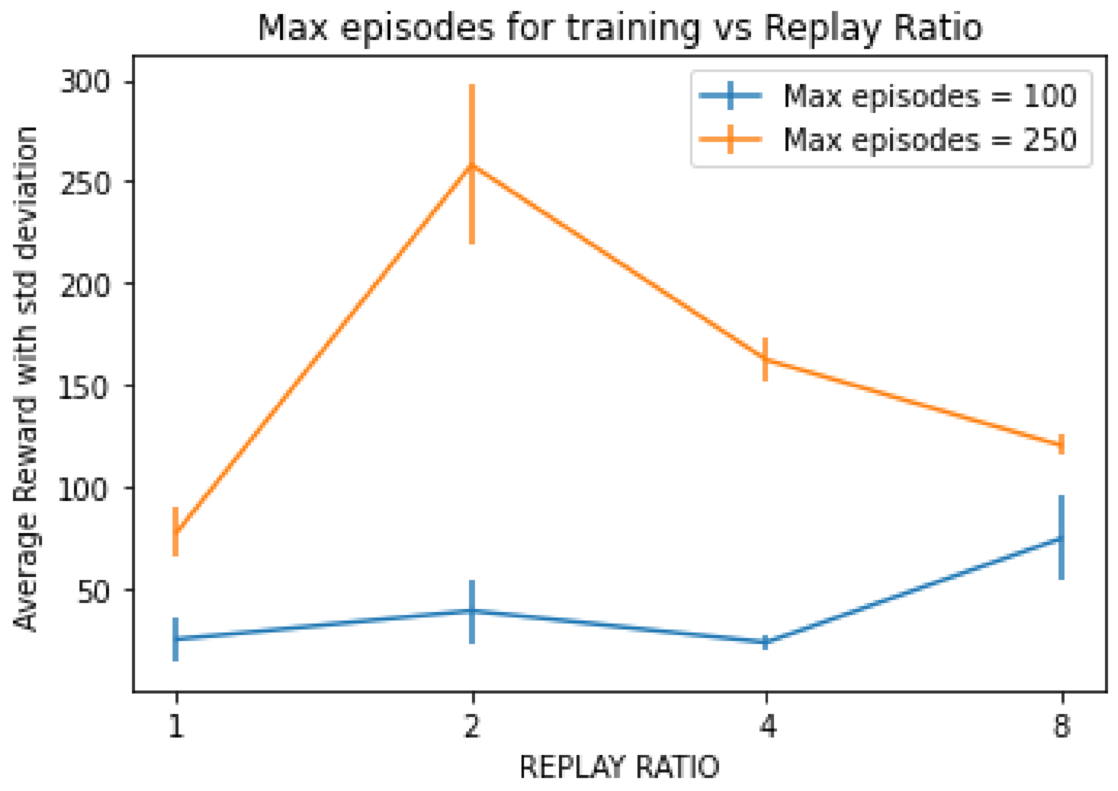
Update Retrace target: $Q^{ret} \leftarrow \bar{\rho}_i\left(Q^{ret} - Q_{\theta'_v}(x_i, a_i)\right) + V_i$

Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.

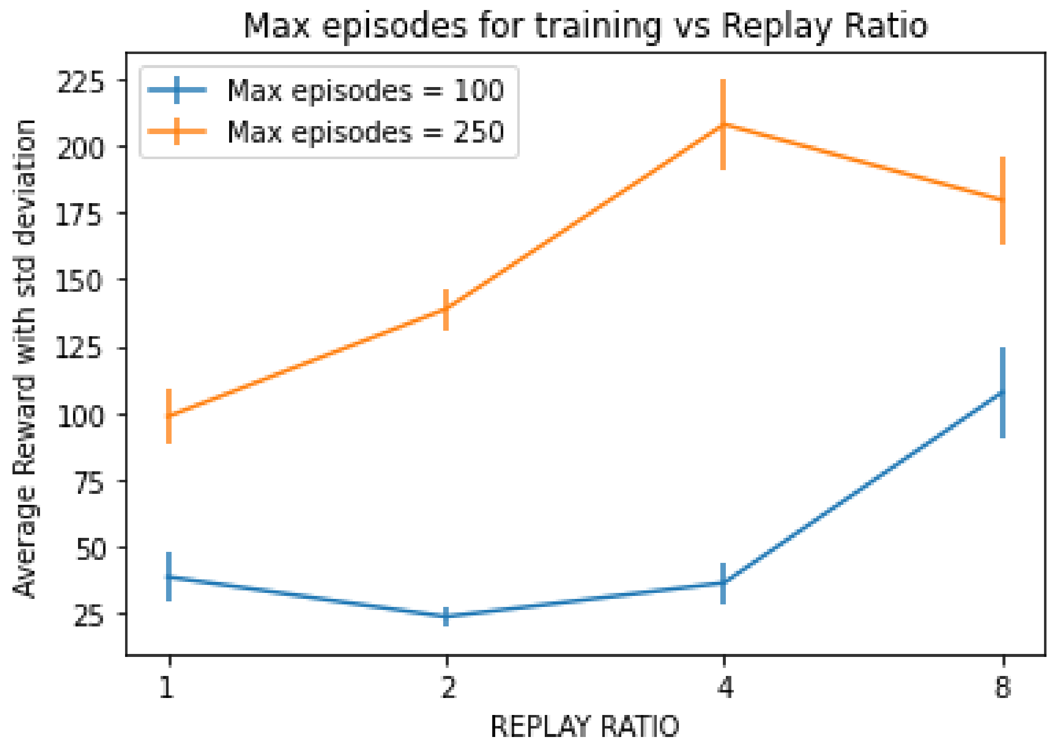
Updating the average policy network: $\theta_a \leftarrow \alpha\theta_a + (1 - \alpha)\theta$

ACER vs ACER with TRPO

Acer without TRPO



Acer with TRPO



Results

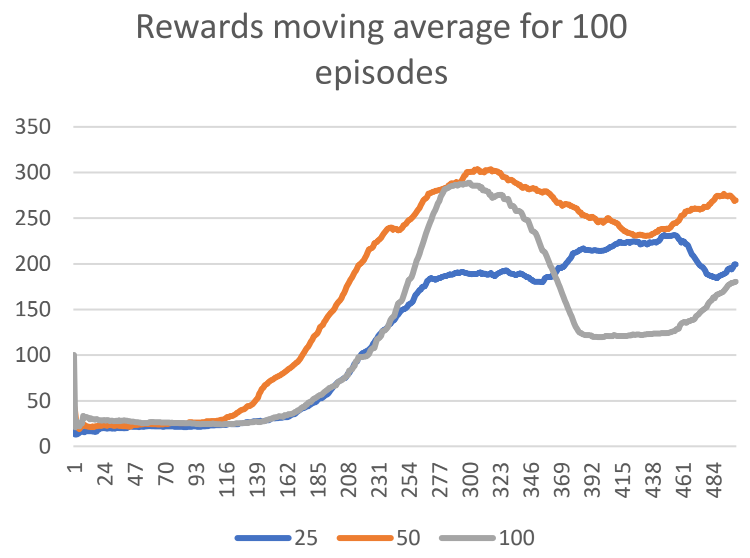
On average, 40 models are trained for each algorithm, and each of these models are tested on 1000 test iterations.

Environment : Cartpole-v1, Max episodes = 500

Algorithms	Average cumulative Reward	Average Deviations in cumulative Reward	Average running time (sec)	Ease of Implementation
Naive Actor-Critic	135.22	45.77	12.34	Simple
Naive Actor-Critic with experience replay	25.34	13.8	20.65	Medium
Actor-Critic Experience replay with marginal Importance sampling	24.57	8.2	38.67	
Actor-Critic Experience replay Retrace	207.5	33.4	40.74	Complex
Actor-Critic Experience replay Retrace with Bias correction	175.83	18.6	45.61	
Actor-Critic Experience replay Retrace with Bias correction and trust region policy optimization	189.13	12.03	46.2	

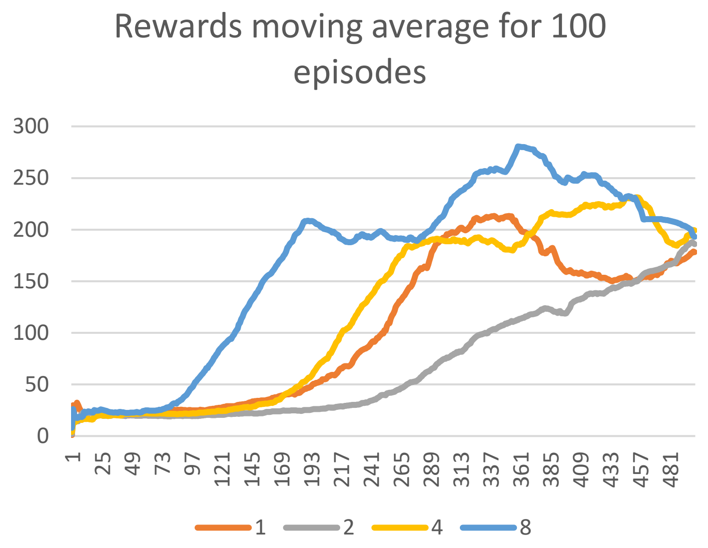
Hyperparameter tuning

REPLAY_BUFFER_SIZE



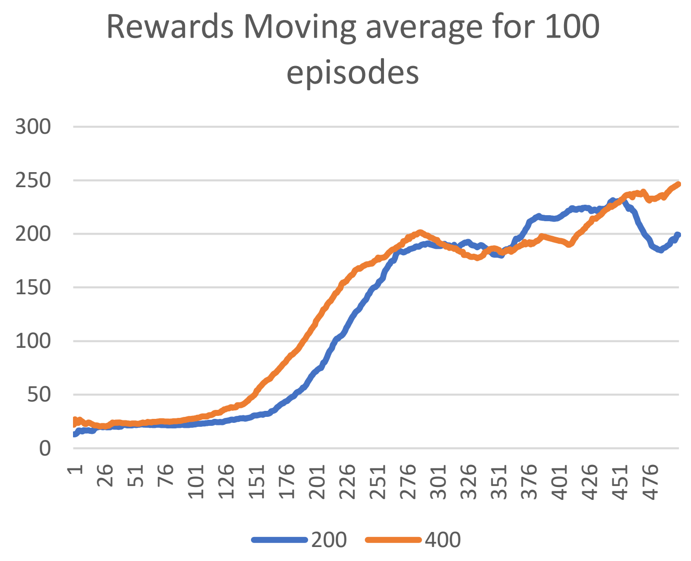
50 is the best

REPLAY_RATIO



8 is the best

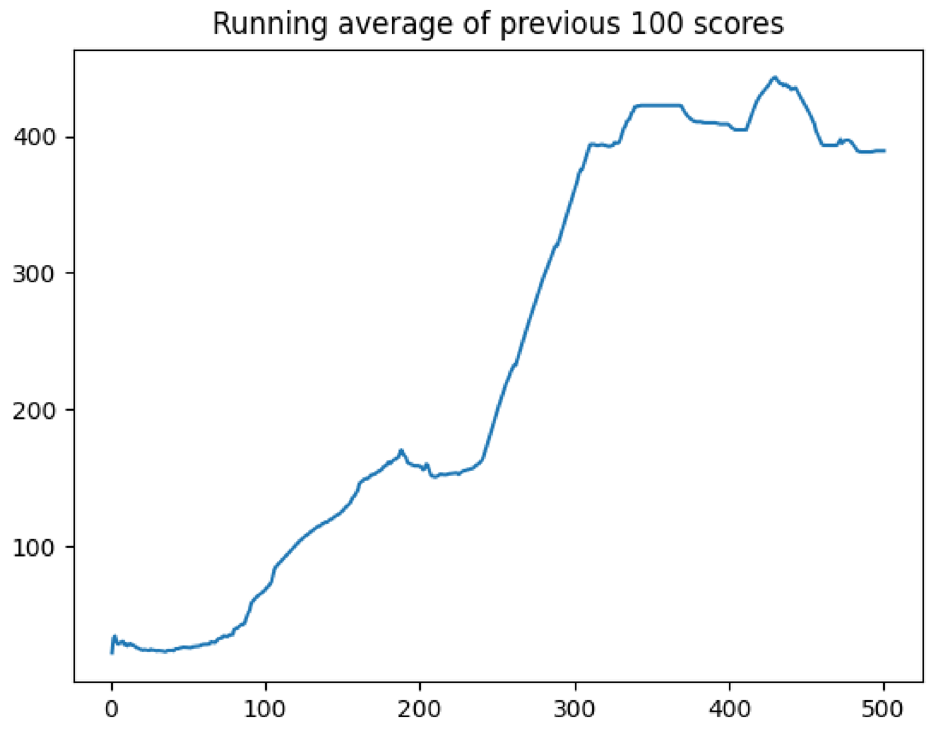
MAX_REPLAY_SIZE



No significant difference

With all hyperparameters at the best

TRAINING



Testing:

- Mean of 1000 episodes: 500
- Std dev. Of 1000 episodes: 0

Conclusion

- We approached the paper by understanding each of the components of ACER algorithm.
- Naïve actor-critic can be improved with importance sampling with bias and retrace, and trust policy optimization
- With hyperparameter tuning, it is possible to achieve maximum rewards, and make the training converge faster.
- Entire Code setup is available at https://www.kaggle.com/code/ronak555/acer1

Next Steps:

- Try to get other hyperparameters tuned and extend the ranges of the current hyperparameters considered.
- Repeat the studies with other environments.

Thank you

Any Questions?