Report and Ablation studies on ACER algorithm

Tina Huang, Ian Lin, Audrey Sung
Department of Information management and finance
National Yang Ming Chiao Tung University
{gyxuan0527.mg07, ianlienfa.mg07, pjs5956.mg07 }@nycu.edu.tw

1 Problem Overview

This paper proposed an algorithm called ACER that combines experience replay and TRPO-like gradient update.

The goal of the problem is to maximize the discounted return $R_t = \sum_{i \geq 0} \gamma^i r_{t+i}$ in expectation

Notation: For the value function: $V^\pi\left(x_t\right) = \mathbb{E}_{a_t}\left[Q^\pi\left(x_t, a_t\right) \mid x_t\right]$

And for the action-value function: $Q^\pi\left(x_t, a_t\right) = \mathbb{E}_{x_{t+1:\infty}, a_{t+1:\infty}}\left[R_t \mid x_t, a_t\right]$

where the actions are determined by policy π

ACER estimates its policy $\pi_\theta(a_t|x_t)$ and value function $V^\pi_{\theta_v}(x_t)$ with deep neural networks.

2 Background and The Algorithm

The original policy gradient is computed as follows:

$$g = E_{x_{0:\infty}, a_{0:\infty}}[\sum_{t \geq 0} A^\pi(x_t, a_t) \nabla_\theta log \pi_\theta(a_t|x_t)]$$

Off-policy learning with experience replay may appear to be an obvious strategy for improving the sample efficiency of actor-critic methods. However, controlling the variance and stability of off-policy estimators is notoriously hard. Importance sampling is one of the most popular approaches for off policy learning. With off-policy sampling, the gradient should be approximate as (Eq 4):

$$g^{marg} = E_{x_t \sim \beta, a_t \sim \mu}[\rho_t \nabla_\theta log \pi_\theta(a_t|x_t) Q^\pi(x_t, a_t)]$$

They adapted behavior policy μ from memory experience as policy and marginal importance weight by importance sampling.

In the following subsection, they adopt the Retrace algorithm to estimate Q^π, propose an importance weight truncation technique to improve the stability of the off-policy actor critic, and introduce a computationally efficient trust region scheme for policy optimization.

Below is the retrace estimator (Eq 5):

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

which is off-policy, low variance and is proven to converge to the value function of the target policy for any behavior policy. Here, $\bar{\rho} = \min\{c, \rho_t\}$, where $\rho_t = \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}$.

We compute Q in Eq(5) using the critic neural network prediction and substitute Q_π in Eq(4) with retrace estimator. Because Retrace is return-based, the advantages of this improvement are to reduce bias in the policy gradient, and to enable faster learning of the critic.

In order to avoid the high variance and bias, the paper decomposed the original equation and implement two methods. First, to deal with the high variance, it clipped the marginal importance weight (ρ_t) into $\bar{\rho}_t = \min(c, \rho_t)$, so that the variance of the gradient estimate is bounded.

However with the gradient clipped, we need a term to preserve the relative magnitude of different gradient Second, the paper added the correction term to ensure the estimate is unbiased:

$$[\frac{\rho_t(a)-c}{\rho_t(a)}]_+$$

So the gradient becomes:

$$g^{marg} = \mathbb{E}_{x_t}[\mathbb{E}_{a_t}[\bar{\rho}_t \bigtriangledown_\theta log\pi_\theta(a_t \mid x_t)Q^\pi(x_t,a_t)] + \mathop{\mathbb{E}}_{a\sim\pi}([\frac{\rho_t(a)-c}{\rho_t(a)}]_+ \bigtriangledown_\theta log\pi_\theta(a \mid x_t)Q^\pi(x_t,a))]$$

Notice that the sampling of (s, a) pair is under the effect of the behavior policy μ. However the truncation has can cause the bias effect on the actual gradient, For example, there will be no difference for ρ with value 10000 or value 1 if c is with value 1. so they introduced a bias term to retain the relative effect of ρ on the gradient The weight of the bias term is designed as:

$$\frac{\rho_t(a)-c}{\rho_t(a)}$$

To milden the effect of large ρ value causing large variance on gradient Then, using the measure called "truncation with bias correction trick" to approximated the $Q^\pi(x,a_t)$ with $Q_{\theta_\nu}(x_t,a)$. Also, approximate the expectation of Markov process by sampling trajectories from the behavior μ. Last, subtracted the baseline $V_{\theta_v}(x_t)$ to reduce variance. Hence, we can rewrite the off-policy ACER gradient as

$$\widehat{g_t}^{acer} = \bar{\rho}_t \bigtriangledown_\theta log\pi_\theta(a_t \mid x)[Q^{ret}(x_t,a_t) - V_{\theta_v}(x_t)] + \mathop{\mathbb{E}}_{a\sim\pi}([\frac{\rho_t(a)-c}{\rho_t(a)}]_+ \bigtriangledown_\theta log\pi_\theta(a \mid x_t)(Q_{\theta_\nu}(x_t,a) - V_{\theta_v}(x_t)))$$

As the above, it turned to updated actor critic by Q estimates, when c = 0. And it turned to updated policy gradient by Retrace, when c = ∞.

In order to reduce high variance of the policy updates in actor-critic, they solve the problem by retricting stepsize. However, the existing method of TRPO is time consuming, they introduced a new TRPO method. The main difference between TRPO and improved TRPO is that they update the policy by running average of past policies which is the average policy network. There are two seperate parts of average policy network output, one is the probability distribution f and the other is the parameters ϕ_θ of distribution f. The relationship between these two terms is : $\phi_\theta : \pi(\cdot \mid x) = f(\cdot \mid \phi_\theta(x))$ Each episode, they update policy by ϕ_θ. The following equation is the ACER policy gradient with respect to ϕ :

$$\begin{aligned}\hat{g}_t^{\text{acer}} =& \bar{\rho}_t \nabla_{\phi_\theta(x_t)} \log f(a_t \mid \phi_\theta(x)) \left[Q^{\text{ret}}(x_t,a_t) - V_{\theta_v}(x_t)\right] \\ &+ \mathop{\mathbb{E}}_{a\sim\pi}\left(\left[\frac{\rho_t(a)-c}{\rho_t(a)}\right]_+ \nabla_{\phi_\theta(x_t)} \log f(a_t \mid \phi_\theta(x)) \left[Q_{\theta_v}(x_t,a) - V_{\theta_v}(x_t)\right]\right)\end{aligned}$$

As for the trust region update, in the first stage, they solved optimization problem by KL divergence :

$$\begin{array}{ll}\underset{z}{\operatorname{minimize}} & \frac{1}{2}\left\|\hat{g}_t^{\text{acer}} - z\right\|_2^2 \\ \text{subject to} & \nabla_{\phi_\theta(x_t)} D_{KL}\left[f(\cdot \mid \phi_{\theta_a}(x_t)) \| f(\cdot \mid \phi_\theta(x_t))\right]^T z \leq \delta\end{array}$$

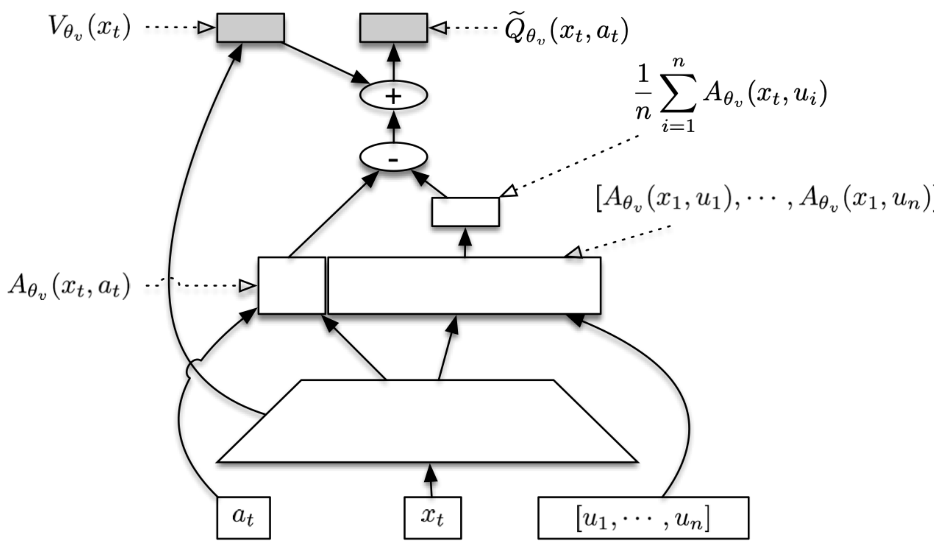
Because the constraint is linear, they don't need to calculate the Hessian. With the advantages, it can solved by the KKT condition easily. Letting $k = \nabla_{\phi_\theta(x_t)} D_{KL}\left[f(\cdot \mid \phi_{\theta_a}(x_t) \| f(\cdot \mid \phi_\theta(x_t)\right]$, the solution is :

$$z^* = \hat{g}_t^{\text{acer}} - \max\left\{0, \frac{k^T\hat{g}_t^{\text{acer}} - \delta}{\|k\|_2^2}\right\} k$$

If the constraint is satisfied, the gradient would not change.

In the second stage, the trust region step is conducted in the space of the statistics of the distribution f rather then the space of the policy parameters. It can successfully avoid additional back-propagation by the policy network.

Figure 1: The Stochastic Deuling Network takes in a state and an action, then output the corresponding value and the action-value with the help of value and advantages



For the continuous case, since it is unpractical to compute value by summing up the action-values, we should deterministically get output from the critic network. To acheive this, the paper uses a Stochastic Dueling Network(SDN) to estimate both V^π and Q^π while maintaining consistency between two estimates.(see Figure 1)

The corresponding estimate for Q^π of SDN is computed as follows:

$$\tilde{Q}_{\theta_v}(x_t, a_t) \sim V_{\theta_v}(x_t) + A_{\theta_v}(x_t, a_t) - \frac{1}{n}\sum_{i=1}^{n} A_{\theta_v}(x_t, u_i), \text{where } u_i \sim \pi(\cdot|x_t)$$

To train the network, the paper as well uses Q^{ret} for target action-value. For the state value V^π, they uses the following target:

$$V^{target}(x_t) = \min\{1, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}\}(Q^{ret}(x_t, a_t) - Q_{\theta_v}(x_t)) + V_{\theta_v}(x_t)$$

, which is derived by applying the trunction and bias correction trick like what have been done on the policy gradient update.

For continous trust region updating, the paper replaced Q^{ret} with Q^{opc}, which is basically Q^{ret} without truncated importance ratio. Though might resulting in a less stable learning, Q^{opc} could better utilize the returns since its not truncated. The choice is made since by experiment they found out that Q^{opc} often leads to faster learning on continous cases.

3 Detailed Implementation

Please explain your implementation in detail. You may do this with the help of pseudo code or a figure of system architecture. Please also highlight which parts of the algorithm lead to the most difficulty in your implementation.

As for the part of ablations, we conduct the following alterations.

1. remove the entropy term
2. remove *trust region constraint*
3. drop the bias term

4. remove the clamping on the actor loss
5. change the *truncation parameter*
6. change Q_opc to Q_ret
7. change Q_ret to Q_opc
8. change the power term of truncated importance weight from $1/d$ to $1/e^d$
9. replace sdn structure to two independent networks

1 5 are conducted under both discrete and continuous environments, while 6 9 are only considered under continuous environment.

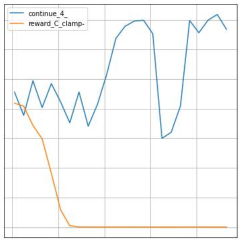
1. Remove the entropy term As we can see, when we remove the entropy term, the performane will get worse.
2. Remove trust region constraint
   In the environment of CartPole and MountainCarContinuous, the removal of trust region constraint from the algorithm seems to be better, especially in the MountainCarContinuous environment.
3. Drop the bias term
   In both environment, the result with bias term dropped turns out to be significantly worse.
4. Remove the clamping on the actor loss
   In both environments, the result gets worse after we remove the clamping function. The reward would change slightly after certain episodes.
5. Change the truncation parameter
   It appears that the results of each corresponding hyperparameters are almost the same in the last few episodes.
6. Change Qopc to Qret
   The result is worse after we conduct the state-action value replacement.
7. Change Qret to Qopc
   The result is also worse after we conduct the state-action value replacement. Moreover, the reward would remain unchanged after some episodes.
8. Change the power term of truncated importance weight from $1/d$ to $1/e^d$
   The change leads to a better performane, since d is the dimensionality of the action space. The alteration causes smaller truncated importane weight.
9. Replace sdn structure to two independent networks
   The result gets worse after replacing the original network to two independent networks. The reward would change slightly after certain episodes.

4 Empirical Evaluation

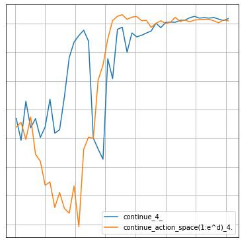
Please showcase your empirical results in this section. Please clearly specify which sets of experiments of the original paper are considered in your report. Please also report the corresponding hyperparameters of each experiment.

1. remove the entropy term 2. remove *trust region constraint* In the environment of CartPole and MountainCarContinuous, the removal of trust region constraint from the algorithm seems to be better, especially in the MountainCarContinuous environment. 3. drop the bias term In both environment, the result with bias term dropped turns out to be significantly worse. 4. remove the clamping on the actor loss 5. change the *truncation parameter* 6. change Q_opc to Q_ret 7. change Q_ret to Q_opc 8. change the power term of truncated importance weight from $1/d$ to $1/e^d$ 9. replace sdn structure to two independent networks

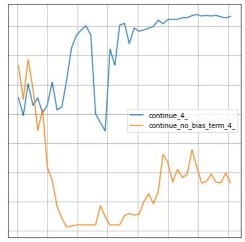
As we can conclude, the algorithm is insensitive to the hyper-parameter tuning.



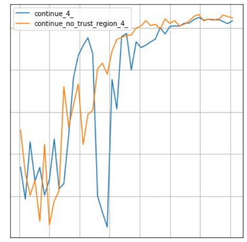
(a) no clamp



(b) $d \to e^d$

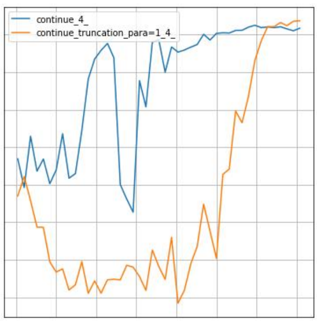


(c) no bias term

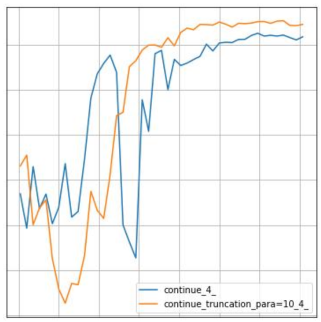


(d) no trust region

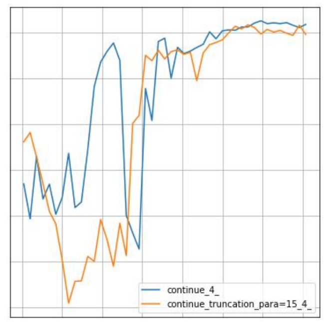
Figure 2: continuous



(a) truncation param $= 1$

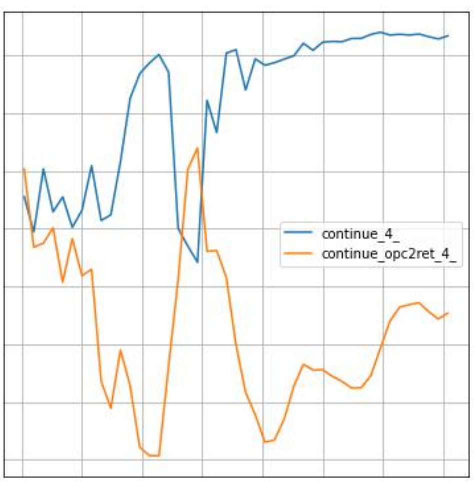


(b) truncation param $= 10$

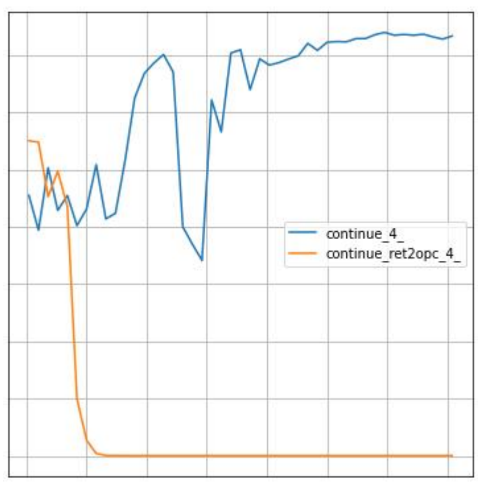


(c) truncation param $= 15$

Figure 3: continous truncation param



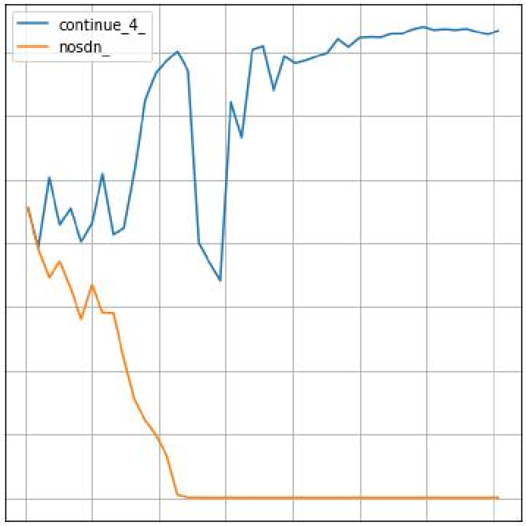
(a) $Q_{opc} \to Q_{ret}$

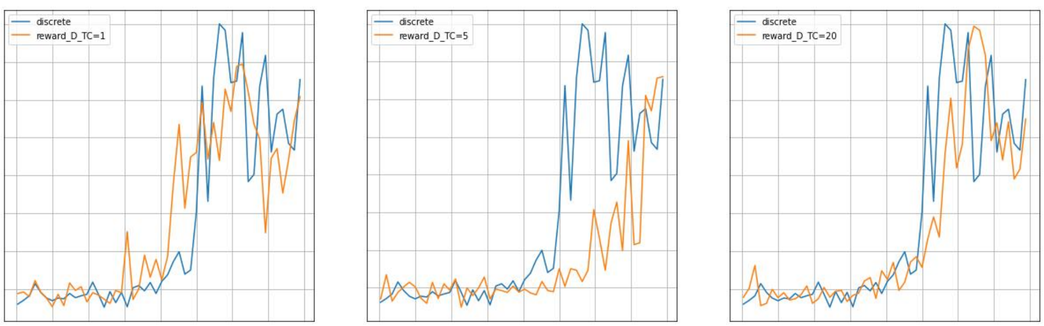


(b) $Q_{ret} \to Q_{opc}$

Figure 4: opc and ret

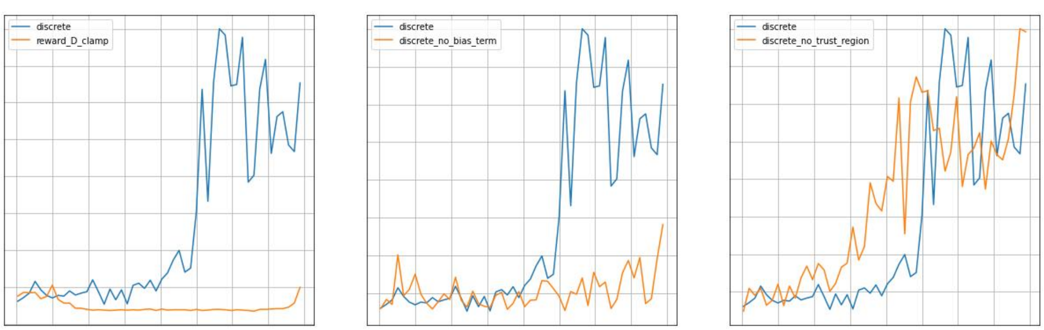
Figure 5: no sdn





(a) truncation param $= 1$ (b) truncation param $= 5$ (c) truncation param $= 20$

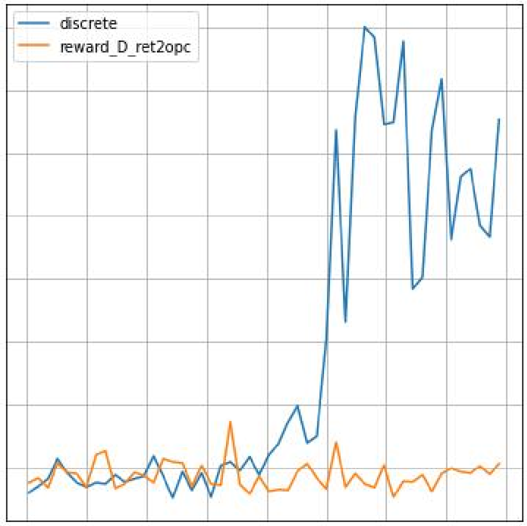
Figure 6: discrete truncation param, default = 10



(a) no clamp (b) no bias term (c) no trust region

Figure 7: discrete

Figure 8: $Q_{ret} \rightarrow Q_{opc}$



5 Conclusion

5.1 latest results on related researches

1. **Refomulating the estimator to provide even smaller variance**
   Including ACER, popular methods in Off-policy learning such as DDPG, TD3, or DPG all have distribution mismatch between the sampling and visitation distribution of updated policy, hence are not provably converge in function approximation settings. Recently, efforts have been made to address the distribution mismatch problem and to reduce the variance. Doubly Robust estimator has been proved to largely reduce variance In Xu et al. [2021], a new doubly robust off-policy actor-critic is proposed to be provably converge with the overall convergence also being doubly robust to the function approximation errors

2. **Allow efficient multi-agent**
   Training RL agent on high-dimensional pixel input is sample inefficient especially in the multi-agent setting since agents not only need to interact with the environment but also with other agents. Su et al. [2022] introduces MASRL, a simple but effective self-supervised task: predicting a learning agent's opponent's future move. In doing this, the agent learns a stronger representation from this additional signal, focusing not only on itself but also on its opponent. By understanding and anticipating the opponent's future moves, MASRL allows the learning agent to develop effective strategies for opponent exploitation. In the paper "COMPETITIVE MULTI-AGENT REINFORCEMENT LEARNING WITH SELF-SUPERVISED REPRESENTATION", MASRL not only stabilizes training, improves sample efficiency, but also allows the agent to generalize and adapt its playing strategy to other unseen expert opponents.

3. **Efficient experience replay**
   We can try to focus on the more efficient experience replay, such as hindsight experience replay and prioritized experience replay, or add some noises when doing sample. Hindsight experience replay Lee and Moon [2021] is referred to a efficiently sampling method which can improve the performance of the deep reinforcement learning by making agent to learn from both failures and successes. Prioritized experience replay Schaul et al. [2015] can also make the deep reinforcement learning algorithm learn more efficiently. Besides, it demonstrates state-of-the-art data efficiency on most of the environment.
   As we known experience replay approaches have an advantage of breaking the temporal correlations in the data achieved by the agent. It causes the data independent and identically distributed (i.i.d). With this advantage, it is better for learning the Q function by using supervised learning approaches such as deep neural networks. That is to say, if the reinforcement learning algorithm is not supervised learning approaches, the effect won't be significant.

In the recent research, there are more and more studies focus on neural networks. However, when they attempt to sequentially learn, they tend to learn the new task while catastrophically forgetting previous ones. The paper Atkinson et al. [2021] applied a model to deal with this problem. The model decomposed memory system into two parts. The first part was continuous learning from reinforcement learning and the second part was a pseudo-rehearsal system that "recalls" items representing previous tasks through a deep generative network. This result showed that it do not need to require additional storage requirements, store raw data or revisit past tasks when the number of tasks increased. Also, it had a better performance.

References

Tengyu Xu, Zhuoran Yang, Zhaoran Wang, and Yingbin Liang. Doubly robust off-policy actor-critic: Convergence and optimality. In *International Conference on Machine Learning*, pages 11581–11591. PMLR, 2021.

DiJia Su, Jason D. Lee, John M. Mulvey, and H. Vincent Poor. Competitive multi-agent reinforcement learning with self-supervised representation. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4098–4102, 2022. doi: 10.1109/ICASSP43922.2022.9747378.

Myoung Hoon Lee and Jun Moon. Deep reinforcement learning-based uav navigation and control: A soft actor-critic with hindsight experience replay approach. *arXiv preprint arXiv:2106.01016*, 2021.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Craig Atkinson, Brendan McCane, Lech Szymanski, and Anthony Robins. Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting. *Neurocomputing*, 428:291–307, 2021.