2022 年度卒業 学士論文

An Enhanced Proximal Policy Optimization Algorithm for Reinforcement Learning

19B30178 **色部 浩男**

指導教員　金森 敬文 教授

February, 2023

東京工業大学　情報理工学院
数理・計算科学系

Abstract

In this thesis, we focus on improving upon a popular deep reinforcement learning algo- rithm known as Proximal Policy Optimization (PPO). PPO is an on-policy actor-critic algorithm that has been widely used. One of its core mechanisms - ratio clipping has demonstrated strong performance in various reinforcement learning tasks. We show how ratio clipping bounds the maximum total variation(TV) distance between policy up- dates, leading to a bound on the approximation error caused by the change in the state distribution.

We then present an inequality that suggests bounding the product of the ratio deviation and its corresponding advantage value. However, simply bounding the product did not result in satisfactory performance. Therefore, we propose a new objective that combines both bounds for improved performance. We call the new algorithm Adaptive clipping PPO(AC-PPO).

AC-PPO is tested on a subset of the 57 Arcade Learning Environments - Atari-5. The experimental results show that AC-PPO outperforms PPO on all five tested games, with smaller KL divergence between policy updates and significantly fewer KL "spikes". On- policy algorithms like PPO typically use only the most recent data to update the policy, discarding the previously collected data as well as old policies. Our results demonstrate the effectiveness of the proposed algorithm in preventing the agent from "overreacting" to signals and unlearning useful policies, a common issue in on-policy algorithms. This study highlights the importance of stable updates in on-policy reinforcement learning algorithms.

As future work, we aim to investigate the reason for failure on the DoubleDunk game, as it may hold key insights for improving our algorithm. We also plan to conduct benchmark testing on other environments. Additionally, we will perform an ablation study to understand the impact of each method on performance, and investigate the algorithm’s sensitivity to hyperparameters.

Contents

Chapter 1

Introduction

Reinforcement learning (RL) is about an agent interacting with the environment, learning an optimal policy, by trial and error, for sequential decision making problems in a wide range of fields in both natural and social sciences, and engineering.[1] In the last several years, deep neural networks have been prevalent in reinforcement learning, leading to breakthroughs in games, robotics, and natural language processing. In 2015, Mnih et al. published the deep Q-network algorithm[2] that can learn to play Atari games from raw pixels. Later in 2019, Five[3] and AlphaStar[4] has shown that RL agents are able to beat human champions in complex e-sports games like Dota 2 and StarCraft II, which present challenges for RL due to long time horizons, partial observability, and high dimensionality of observation and action spaces[3].

In this thesis, we focus on improving upon a popular deep reinforcement learning algorithm known as Proximal Policy Optimization (PPO). PPO is an on-policy actor-critic algorithm that has been widely used. One of its core mechanisms - ratio clipping has demonstrated strong performance in various reinforcement learning tasks. In the following sections, we show how ratio clipping bounds the maximum total variation(TV) distance between policy updates, leading to a bound on the approximation error caused by the change in the state distribution. We then present an inequality that suggests bounding the product of the ratio deviation and its corresponding advantage value. However, simply bounding the product did not result in satisfactory performance. Therefore, we propose a new objective that combines both bounds for improved performance. We call the new algorithm Adaptive clipping PPO(AC-PPO). AC-PPO is tested on a subset of the 57 Arcade Learning Environments[5] - Atari-5[6] and the results show that the proposed algorithm outperforms the baseline.

Chapter 2

Preliminaries

2.1 Discounted Markov Devision Processes

In reinforcement learning, the interactions between the agent and the environment are often modeled by an discounted Markov Decision Process(MDP), which is a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \mu)$, where

- $\mathcal{S}$ is the set of all possible states. We will assume that $\mathcal{S}$ is finite.
- $\mathcal{A}$ is the set of all possible actions. Samely, we wil assume that $\mathcal{A}$ is finite.
- $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a transition function. $p(s'|s, a)$ is the probability of transitioning into s' upon taking action a in state s.
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function. $r(s, a)$ is the expcted reward upon taking action a at action s.
- $\gamma \in [0, 1]$ is a discount factor that trades off later rewards to ealier ones.
- $\mu \in \Delta(\mathcal{S})$ is an initial state distribution where the initial state s_0 is generated.

2.2 The goal, policies, values, advantages and visitation distribution

Policies. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ defines the behavior of the agent in the environment. $\pi(a|s)$ is the probability of the agent taking action a observing state s.

Goal. The goal of a agent is to learn a policy π that maximizes the following objective, namely the expected total discounted reward:

$$J(\pi) = \mathop{\mathbb{E}}_{s_0, a_0, \ldots} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \tag{2.1}$$

$$\text{where} \quad s_0 \sim \mu, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t).$$

Values. Now we define the value function $V_\pi : \mathcal{S} \to \mathbb{R}$ as the objective above starting

from a specific state s:

$$V_\pi(s) = \mathop{\mathbb{E}}_{s_0,a_0,\dots}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \tag{2.2}$$
$$\text{where} \quad s_0 = s, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t,a_t).$$

And we have the following relationship between $J(\pi)$ and V_π:

$$J(\pi) = \mathop{\mathbb{E}}_{s_0\sim\mu}[V_\pi(s_0)] \tag{2.3}$$

Similarly, the Q-value(action-state) function $Q_\pi : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$ is defined as

$$Q_\pi(s,a) = \mathop{\mathbb{E}}_{s_0,a_0,\dots}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \tag{2.4}$$
$$\text{where} \quad s_0 = s, a_0 = a, s_{t+1} \sim p(\cdot|s_t,a_t), a_t \sim \pi(\cdot|s_t).$$

Advantages. We define an advantage function $A_\pi : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$ to measure how much better a given action is compared to the average action under a certain policy:

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s). \tag{2.5}$$

Note that for each $s\in\mathcal{S}$,

$$\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}[A_\pi(s,a)] = \mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}[Q_\pi(s,a) - V_\pi(s)] = V_\pi(s) - V_\pi(s) = 0. \tag{2.6}$$

Visitation distribution. We may want to measure the frequency of the agent visiting each state over a time horizon, namely discounted state visitation distribution. First we define the state transition probability $p_\pi : \mathcal{S}\to\Delta(\mathcal{S})$:

$$p_\pi(s'|s) := \sum_a \pi(a|s)p(s'|s,a). \tag{2.7}$$

Denote the corresponding transition matrix by P_π. And we write the objective $J(\pi)$ explicitly as following:

$$J(\pi) = \mathop{\mathbb{E}}_{\substack{s_0\sim\mu\\ a_0\sim\pi(\cdot|s_0)}}[r(s_0,a_0)] + \gamma \mathop{\mathbb{E}}_{\substack{s_1\sim P_\pi\mu\\ a_1\sim\pi(\cdot|s_1)}}[r(s_1,a_1)] + \gamma^2 \mathop{\mathbb{E}}_{\substack{s_2\sim P_\pi^2\mu\\ a_2\sim\pi(\cdot|s_2)}}[r(s_2,a_2)] + \dots. \tag{2.8}$$

Now if we define the discounted state visitation distribution:

$$d_\pi := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P_\pi^t \mu, \tag{2.9}$$

and we can write

$$J(\pi) = \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{\substack{s\sim d_\pi\\ a\sim\pi(\cdot|s)}}[r(s,a)]. \tag{2.10}$$

Note that $s\sim d_\pi, a\sim\pi(\cdot|s)$ may be denoted as $(s,a)\sim d_\pi$ for simplicity.

2.3 Performance difference and policy optimization

The differences of performance between two policies can be expressed as following:

Lemma 2.3.1. [7]

$$J(\tilde{\pi}) - J(\pi) = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim d_{\tilde{\pi}}} [A_\pi(s,a)] \tag{2.11}$$

Proof.

$$\begin{aligned}
\frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim d_{\tilde{\pi}}} [A_\pi(s,a)] &= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim d_{\tilde{\pi}}} [Q_\pi(s,a) - V_\pi(s)] \\
&= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim d_{\tilde{\pi}}} \left[r(s,a) + \gamma \mathop{\mathbb{E}}_{s'\sim p_{\tilde{\pi}}(\cdot|s)} \left[V_\pi(s') \right] - V_\pi(s) \right] \\
&= \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{(s,a)\sim d_{\tilde{\pi}}} [r(s,a)] + \frac{1}{1-\gamma} \left(\gamma \mathop{\mathbb{E}}_{s'\sim P_{\tilde{\pi}} d_{\tilde{\pi}}} \left[V_\pi(s') \right] - \mathop{\mathbb{E}}_{s\sim d_{\tilde{\pi}}} [V_\pi(s)] \right) \\
&= J(\tilde{\pi}) + \gamma \sum_{t=0}^{\infty} \gamma^t \mathop{\mathbb{E}}_{s'\sim P_{\tilde{\pi}}^{t+1}\mu} \left[V_\pi(s') \right] - \sum_{t=0}^{\infty} \gamma^t \mathop{\mathbb{E}}_{s\sim P_{\tilde{\pi}}^{t}\mu} [V_\pi(s)] \\
&= J(\tilde{\pi}) + \sum_{t=0}^{\infty} \gamma^{t+1} \mathop{\mathbb{E}}_{s'\sim P_{\tilde{\pi}}^{t+1}\mu} \left[V_\pi(s') \right] - \sum_{t=0}^{\infty} \gamma^t \mathop{\mathbb{E}}_{s\sim P_{\tilde{\pi}}^{t}\mu} [V_\pi(s)] \\
&= J(\tilde{\pi}) - \mathop{\mathbb{E}}_{s\sim\mu} [V_\pi(s)] \\
&= J(\tilde{\pi}) - J(\pi).
\end{aligned}$$

□

Suppose our current policy is π, this lemma helps us to find a improved policy $\tilde{\pi}$ by optimizing the right hand side. However, it is usually difficult to directly do so, due to its dependency on the distribution of target policy $d_{\tilde{\pi}}$. Here we consider the approximation

$$\mathop{\mathbb{E}}_{(s,a)\sim d_{\tilde{\pi}}} [A_\pi(s,a)] \approx \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\tilde{\pi}}} [A_\pi(s,a)] = \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\tilde{\pi}}} \left[\frac{\tilde{\pi}(a|s)}{\pi(a|s)} A_\pi(s,a) \right]. \tag{2.12}$$

The approximation error is caused by the change of state distribution. A bound is provided by the following theorem.

Theorem 2.3.2. [8] Let $\mathcal{E} = \mathop{\mathbb{E}}_{(s,a)\sim d_{\tilde{\pi}}} [A_\pi(s,a)] - \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\tilde{\pi}}} [A_\pi(s,a)]$. Define $D_{\mathrm{TV}}^{\mathrm{MAX}}(\pi,\tilde{\pi}) = \max_s D_{\mathrm{TV}}(\pi(\cdot|s),\tilde{\pi}(\cdot|s))$ where D_{TV} is the total variance(TV) distance. Then

$$|\mathcal{E}| \le \frac{4\gamma\alpha^2\xi}{1-\gamma} \quad \text{where } \xi = \max_{s,a} |A_\pi(s,a)| \text{ and } \alpha = D_{\mathrm{TV}}^{\mathrm{MAX}}(\pi,\tilde{\pi}). \tag{2.13}$$

2.4 PPO objective

Proximal Policy Optimization(PPO)[9] is a widely used reinforcement learning algorithm proposed by Schulman et al. It optimizes the following objective*1: let $\rho_{a|s} = \frac{\tilde{\pi}(a|s)}{\pi(a|s)}$,

$$L^{PPO} = \mathop{\mathbb{E}}_{(s,a)\sim d_\pi}\left[\min\left(\rho_{a|s}\hat{A}_\pi(s,a), \text{clip}(\rho_{a|s}, 1-\epsilon, 1+\epsilon)\hat{A}_\pi(s,a)\right)\right]. \tag{2.14}$$

The first term is the approximated objective in (2.12). And the second term, $\text{clip}(\rho_{a|s}, 1-\epsilon, 1+\epsilon)\hat{A}_\pi(s,a)$ uses a hyperparameter ϵ to limit the ratio deviation $|\rho - 1|$ by removing the incentive for ρ moving outside of $[1-\epsilon, 1+\epsilon]$. Then the minimum of the unclipped and clipped objective is taken, so that the final objective is a lower bound on the former.

As we see in the following proposition, ratio clipping can effectively bound the maximum TV divergence appeared in (2.13).

Proposition 2.4.1. Suppose $1-\epsilon \leq \frac{\tilde{\pi}(a|s)}{\pi(a|s)} \leq 1+\epsilon$ for all (a,s). Then the following bound holds: $D_{\text{TV}}^{\text{MAX}}(\pi, \tilde{\pi}) \leq \epsilon/2$.

Proof. For every state $s \in \mathcal{S}$, we have

$$D_{\text{TV}}(\pi(\cdot|s), \tilde{\pi}(\cdot|s)) = \frac{1}{2}\sum_{a\in\mathcal{A}} |\tilde{\pi}(a|s) - \pi(a|s)| \leq \frac{1}{2}\sum_{a\in\mathcal{A}} \epsilon\pi(a|s) \leq \frac{\epsilon}{2}.$$

□

2.5 Generalized Advantage Estimator

Now we discuss a method of estimating advantage values called Generalized Advantage Estimator(GAE)[10]. It has shown strong performance and has become a default choice for advantage estimation[11].

Suppose V is an estimator for the true value function V_π. Using V, we are able to derive a class of advantage function estimators by n-step bootstrapping[12] as follows:

$$\begin{aligned}
\hat{A}_t^{(1)} &= r_t + \gamma V(s_{t+1}) - V(s_t) =: \delta_t \\
\hat{A}_t^{(2)} &= r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t) = \delta_t + \gamma\delta_{t+1} \\
&\vdots \\
\hat{A}_t^{(\infty)} &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots - V(s_t) = \delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \ldots
\end{aligned}$$

*1 Advantage function/values are estimated as they are usually unknown. This is discussed in the following section.

where we refer to δ_t as the temporal difference[12] residual at timestep t. Now we face a tradeoff when utilizing these estimators - $\hat{A}_t^{(k)}$ with small k has low variance but high bias, those with large k have low bias but high variance. Instead of choosing a specific k, we consider using exponentially-decayed average of all possible k-step TD estimators $\hat{A}_t^k$:

$$\begin{aligned}
\hat{A}_t^{GAE(\gamma,\lambda)} &= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t{}^{(2)} + \lambda^2 \hat{A}_t{}^{(3)} + \dots\right) \\
&= (1-\lambda)\left(\delta_t + \lambda(\delta_t + \gamma\delta_{t+1}) + \dots\right) \\
&= (1-\lambda)\left(\delta_t(1+\lambda+\lambda^2+\dots) + \gamma\delta_{t+1}(\lambda+\lambda^2+\dots) + \dots\right) \\
&= (1-\lambda)\left(\delta_t \frac{1}{1-\lambda} + \gamma\delta_{t+1}\frac{\lambda}{1-\lambda} + \dots\right) \\
&= \sum_{l=0}^{\infty} (\lambda\gamma)^l \delta_{t+l}.
\end{aligned}$$

Now the trade-off between bias and variance is controlled by parameter γ. Note that the relationship between adjacent advantages(in terms of timestep) is given by

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \delta_t + \gamma\lambda \hat{A}_{t+1}^{GAE(\gamma,\lambda)}.$$

2.6 Algorithm

PPO is an on-policy actor-critic[13] algorithm. On-policy algorithms run the current policy in the environment, evaluate and update the policy based on the observed rewards and next states. By contrast, off-policy algorithms may use a unrelated behavioural policy to collect the data and evaluate policies. Actor-critic algorithms use two models: one to learn the policy(actor) and the other to approximate and learn action-state(or state) values(critic). An high-level overview of on-policy actor-critic algorithm is shown as follows:

Algorithm 1 On-policy actor-critic algorithms

Initialize the policy π and value functions V(or Q) approximated by parameters θ and w respectively.
for iteration $= 1, 2, \dots$ **do**
  Run policy π for certain steps and collect data $\mathcal{D}$.
  for epoch $= 1, 2, \dots$ **do**
    Using $\mathcal{D}$, update θ and w by optimizing loss functions L_v and L_π respectively.
  end for
end for

The full detailed pseudo code of PPO algorithm is shown in AppendixB.

Chapter 3

Proposed Method

3.1 Adaptive Clipping PPO

PPO objective tries to limit the probability ratio deviation $\left|\frac{\tilde{\pi}(a|s)}{\pi(a|s)} - 1\right|$ for all rollout (a, s) during updates. However, the following bound on approximation error suggests that it should also be helpful to bound the product of the ratio deviation and its corresponding advantage value.

Theorem 3.1.1. Let $\mathcal{E} = \mathop{\mathbb{E}}_{(s,a)\sim d_{\tilde{\pi}}}[A_\pi(s,a)] - \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\tilde{\pi}}}[A_\pi(s,a)]$. Then the following bound holds:

$$|\mathcal{E}| \leq 2\delta \quad \text{where } \delta = \max_{s,a}\left|\left(\frac{\tilde{\pi}(a|s)}{\pi(a|s)} - 1\right) A_\pi(s,a)\right|,$$

Proof.

$$\begin{aligned}
\mathcal{E} = \mathop{\mathbb{E}}_{(s,a)\sim d_{\tilde{\pi}}}[A_\pi(s,a)] - \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\tilde{\pi}}}[A_\pi(s,a)] &= \sum_{s\in\mathcal{S}} d_{\tilde{\pi}}(s) \sum_{a\in\mathcal{A}} \tilde{\pi}(a|s) A_\pi(s,a) - \sum_{s\in\mathcal{S}} d_\pi(s) \sum_{a\in\mathcal{A}} \tilde{\pi}(a|s) A_\pi(s,a) \\
&= \sum_{s\in\mathcal{S}} \left((d_{\tilde{\pi}}(s) - d_\pi(s)) \sum_{a\in\mathcal{A}} \tilde{\pi}(a|s) A_\pi(s,a) \right) \\
&= \sum_{s\in\mathcal{S}} \left((d_{\tilde{\pi}}(s) - d_\pi(s)) \left(\sum_{a\in\mathcal{A}} \tilde{\pi}(a|s) A_\pi(s,a) - \underbrace{\sum_{a\in\mathcal{A}} \pi(a|s) A_\pi(s,a)}_{0} \right) \right) \\
&= \sum_{s\in\mathcal{S}} \left((d_{\tilde{\pi}}(s) - d_\pi(s)) \underbrace{\sum_{a\in\mathcal{A}} \pi(a|s)}_{1} \underbrace{\left(\frac{\tilde{\pi}(a|s)}{\pi(a|s)} - 1\right) A_\pi(s,a)}_{|\cdot|\leq\delta} \right)
\end{aligned}$$

.

Taking $|\cdot|$ on both sides and we get

$$\begin{aligned}|\mathcal{E}| &\leq \delta \sum_{s\in\mathcal{S}} \left| d_{\tilde{\pi}(s)} - d_{\pi}(s) \right| \\ &= 2\delta \sum_{\substack{s\in\mathcal{S} \\ d_{\tilde{\pi}(s)} \geq d_{\pi}(s)}} (d_{\tilde{\pi}}(s) - d_{\pi}(s)) \\ &\leq 2\delta \sum_{s\in\mathcal{S}} d_{\tilde{\pi}}(s) = 2\delta.\end{aligned}$$

□

Despite the obvious drawback that the scale of advantages can vary across different environments, it has become a common practice to normalize the advantages before they are used in the objective function. This normalization technique has been shown to yield improved empirical performance and can be found in implementations of popular libraries[14, 15, 16]. Bounding the fore-mentioned product, we propose the following objective:[*1]

$$L^{\times} = \mathop{\mathbb{E}}_{(s,a)\sim d_{\pi}} \left[\min \left((\rho_{a|s} - 1)\hat{A}(s,a), \delta \right) \right]$$

where $\rho_{a|s} = \frac{\tilde{\pi}(a|s)}{\pi(a|s)}$. However, our experimental results indicate that this approach does not perform well. One potential explanation for this is that for small advantages, the ratio deviation may still be significant. To address this issue, we propose an alternative objective that aims to bound both the ratio deviation and its product with the normalized advantage:

$$L^{\text{AC-PPO}} = \mathop{\mathbb{E}}_{(s,a)\sim d_{\pi}} \left[\min \left(\rho_{a|s}\hat{A}(s,a), \text{clip}(\rho_{a|s}, 1 - \frac{\epsilon}{1 + c|\hat{A}(s,a)|}, 1 + \frac{\epsilon}{1 + c|\hat{A}(s,a)|})\hat{A}(s,a) \right) \right].$$

Hyperparameter c is introduced to determine the weight between two bounds. When $c = 0$, the objective is identical to PPO. And when $c \to \infty$, the objective becomes equivalent to $L^{\times}$. In our experiment, we heuristically choose $c = 1$ through several hyperparameter sweeps. We call the modified PPO algorithm using the objective above by adaptive clipping PPO(AC-PPO).

[*1] Note that since $\mathop{\mathbb{E}}_{(s,a)\sim d_{\pi}}[A(s,a)] = 0$(2.2), the first term $(\rho_{a|s} - 1)\hat{A}(s,a)$ is the same in expectation with the original $\rho_{a|s}\hat{A}(s,a)$.

Chapter 4

Experiment

We first compare our proposed method with PPO on a subet of the 57 Arcade Learning Environments[5] - Atari-5[6], due to the limited computational resources. The set of five specific games has been reported to produce median score estimates within 10% of their true values [6]. We use CleanRL[15] which is a popular open-source library as the baseline implementation. The choice of hyperparameters follows the original paper[9] which is shown in AppendixA. All the results are reported by the average of most recent 100 episodic returns with their means and standard deviations, using 3 different random seeds.

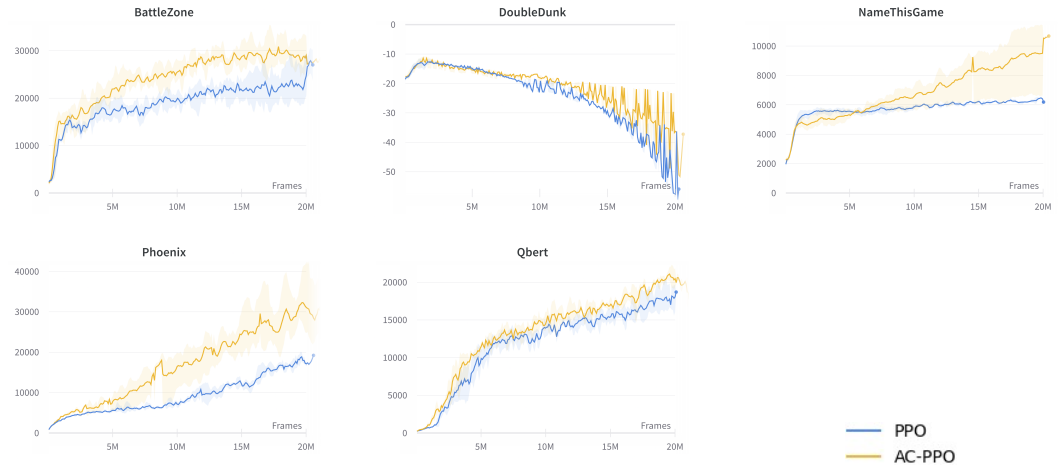

Fig. 4.1: Comparison of average episodic returns between AC-PPO and the baseline on Atari-5 benchmarks, training for 20M frames. Shading indicates standard deviation over 3 random seeds.

We see that AC-PPO outperforms PPO on all five environments despite both failing on learning DoubleDunk in 20M frames of training. We further investigates how much the policy changes with each update by estimating KL divergence between policy updates. It is estimated using rollout samples given by $\mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\left[\log \frac{\pi(a|s)}{\pi_{\text{new}}(a|s)}\right]$ where $\mathcal{B}$

is the previous rollout data.

In order to conduct a fair comparison, we aimed to make the average clipping ratio of AC-PPO as similar as possible to that of the baseline PPO algorithm:

$$\mathop{\mathbb{E}}_{(s,a)\sim d_\pi}\left[\frac{\epsilon_{\text{AC-PPO}}}{1+|\hat{A}(s,a)|}\right] \approx \epsilon_{\text{PPO}}.$$

We chose accordingly $\epsilon_{\text{AC-PPO}} = 0.16 (= 1.6\epsilon_{\text{PPO}})$ and the resulting average clipping ratio is shown in AppendixC. As observed in the following, AC-PPO exhibited a smaller KL-divergence, which can be interpreted as a more conservative policy update. Despite the smaller updates, AC-PPO performed better. Additionally, it exhibited significantly fewer KL "spikes", which are large updates to the policy that can lead to the agent forgetting useful policies. This can be attributed to the higher stabilization of the updates by combining the ratio clipping and the proposed ratio-advantage-product clipping.

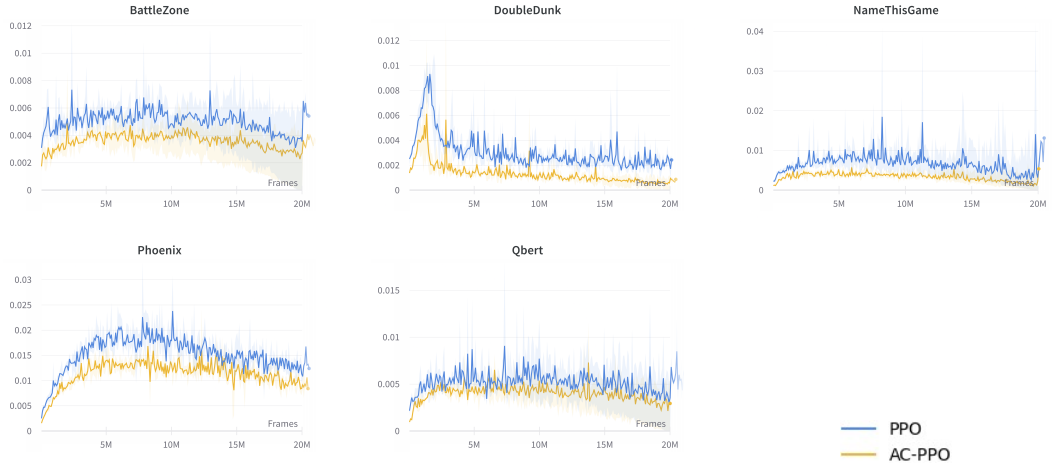

Fig. 4.2: Comparison of KL divergence between policy updates

Chapter 5

Conclusion

In conclusion, this thesis presents an improvement to the Proximal Policy Optimization (PPO) algorithm. By combining ratio clipping with a new objective that bounds the product of the ratio deviation and its corresponding advantage value, we present Adaptive Clipping PPO (AC-PPO). The experimental results show that AC-PPO outperforms PPO on all five tested games, with smaller KL divergence between policy updates and significantly fewer KL "spikes". On-policy algorithms like PPO typically use only the most recent data to update the policy, discarding the previously collected data as well as old policies. These results demonstrate the effectiveness of the proposed algorithm in preventing the agent from "overreacting" to signals and unlearning useful policies, a common issue in on-policy algorithms. This study highlights the importance of stable updates in on-policy reinforcement learning algorithms.

As future work, we aim to investigate the reason for failure on the DoubleDunk game(unfortunately videos of the agent's gameplay were not available for investigating in our experiment), as it may hold key insights for improving our algorithm. We also plan to conduct benchmark testing on other environments, such as Mujoco[17] which is a robotic control environment and Procgen[18] which is a procedurally generated game environment. Additionally, we will perform an ablation study to understand the impact of each method on performance, and investigate the algorithm's sensitivity to hyperparameters.

A

Hyperparameters

Hyperparameter	Value
Learning rate	$\alpha \times 2.5 \times 10^{-4}$ (linearly decaying)
Parallel environments	8
Rollout horizon	128
Discount rate γ	0.99
GAE λ	0.95
PPO ϵ	0.1
Entropy bonus coef	0.01
Update epochs	4
Mini-batch size	256
Gradient clipping	0.5

B

PPO algorithm details

Algorithm 2 PPO Algorithm

Initialize parameters θ for approximating policy π and value function v.
Initialize Adam optimizer O for θ.
Initialize next observation $o_{\text{next}} = E.\text{reset}$
Create vectorized environment E containing N parallel environments

for iteration$= 1, 2, \ldots, I$ **do**
  Allocate $\mathcal{D} = \{ (o[i], a[i], \pi(a|o)[i], r[i], d[i], v[i], A[i]) \}_{i=1}^{T}$.

  ▷ Rollout Phase
  for $t = 1, \ldots, T$ **do**
    Store $o[t] = o_{\text{next}}, d[t] = d_{\text{next}}$
    Get $a[t] \sim \pi(\cdot|o[t])$ and $v[t] = v(o[t])$
    Store $\pi(a|o)[i] = \pi(a[i]|o[i])$
    Step simulator: $o_{\text{next}}, r[t], d_{\text{next}} = E.\text{step}(a[t])$
  end for

  ▷ Learning Phase
  ▷ Compute GAE
  for $t = T - 1, \ldots, 1$ **do**
    $\delta = r[t] + \gamma v[t+1] - v[t]$
    $A[t] = \delta + \gamma\lambda A[t+1]$
  end for
  Let TD(λ) return $R = A + v$
  Prepare the batch $\mathcal{B} = \mathcal{D}, A, R$ and flatten $\mathcal{B}$
  ▷ Optimize the surrogate with minibatch SGD
  for epoch$= 1, 2, \ldots, K$ **do**
    for mini-batch $\mathcal{M}$ of size m in $\mathcal{B}$ **do**
      Normalize advantage $\mathcal{M}.A = \frac{\mathcal{M}.A - \mathcal{M}.A.\text{mean}()}{\mathcal{M}.A.\text{std}() + 10^{-8}}$
      Let ratio $r = \frac{\pi(M.a|M.o)}{M.\pi(a|o)}$
      Let $L_\pi = \min(r\mathcal{M}.A, \text{clip}(r, 1-\epsilon, 1+\epsilon)\mathcal{M}.A)$
      Let $L_V = \text{MSE}(\mathcal{M}.R, v(\mathcal{M}.o))$
      Let $L_S = \pi(\mathcal{M}.o).\text{entropy}()$
      Back-propagate loss $L = -L_\pi + c_1 L_V - c_2 L_S$
      Step the optimizer O to initiate gradient descent
    end for
  end for
end for

C

Average clipping ratio

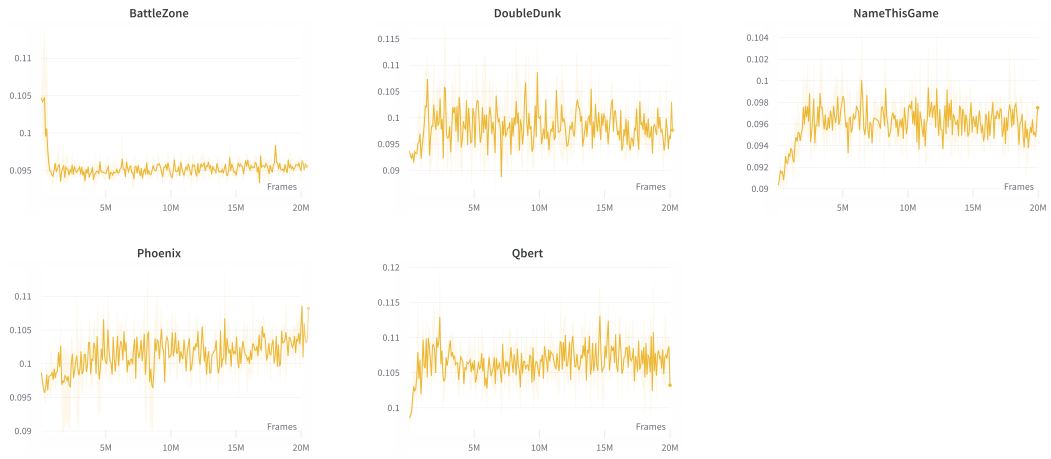

Fig. C.1: Average clipping ratio of AC-PPO

Acknowledgements

I would like to express my deep gratitude to my supervisor, Prof. Kanamori, for providing support and guidance throughout the year. I appreciate his efforts in organizing meetings where I received insightful suggestions and information that helped me accomplish this work. Also I am grateful for the generous support in covering the costs for using cloud computing services for my research.

I would also like to extend my gratitude to all members of Kanamori Lab as well as my family and friends who have provided me with support, and encouragement throughout this journey.

Bibliography

[1] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[4] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[5] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[6] M. Aitchison, P. Sweetser, and M. Hutter, "Atari-5: Distilling the arcade learning environment down to five games," *arXiv preprint arXiv:2210.02019*, 2022.

[7] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *In Proc. 19th International Conference on Machine Learning*, Citeseer, 2002.

[8] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[11] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, *et al.*, "What matters for on-policy deep actor-critic methods? a large-scale study," in *International conference on learning representations*, 2021.

[12] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction. 2nd," *Manuscript in preparation*, 2018.

[13] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and

K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.

[14] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines." `https://github.com/openai/baselines`, 2017.

[15] S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. Araújo, "Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms," *Journal of Machine Learning Research*, vol. 23, no. 274, pp. 1–18, 2022.

[16] M. W. Hoffman, B. Shahriari, J. Aslanides, G. Barth-Maron, N. Momchev, D. Sinopalnikov, P. Stańczyk, S. Ramos, A. Raichuk, D. Vincent, L. Hussenot, R. Dadashi, G. Dulac-Arnold, M. Orsini, A. Jacq, J. Ferret, N. Vieillard, S. K. S. Ghasemipour, S. Girgin, O. Pietquin, F. Behbahani, T. Norman, A. Abdolmaleki, A. Cassirer, F. Yang, K. Baumli, S. Henderson, A. Friesen, R. Haroun, A. Novikov, S. G. Colmenarejo, S. Cabi, C. Gulcehre, T. L. Paine, S. Srinivasan, A. Cowie, Z. Wang, B. Piot, and N. de Freitas, "Acme: A research framework for distributed reinforcement learning," *arXiv preprint arXiv:2006.00979*, 2020.

[17] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033, IEEE, 2012.

[18] S. Mohanty, J. Poonganam, A. Gaidon, A. Kolobov, B. Wulfe, D. Chakraborty, G. Šemetulskis, J. Schapke, J. Kubilius, J. Pašukonis, *et al.*, "Measuring sample efficiency and generalization in reinforcement learning benchmarks: Neurips 2020 procgen benchmark," *arXiv preprint arXiv:2103.15332*, 2021.