Partial Advantage Estimator for Proximal Policy Optimization

Yizhao Jin* , Xiulei Song*, Gregory Slabaugh, Simon Lucas

***Abstract*—This paper proposes an innovative approach to the Generalized Advantage Estimator (GAE) to address the bias-variance trade-off in truncated roll-outs during reinforcement learning. In typical GAE implementations, the k-step advantage is estimated using a lambda-weighted average, until the terminal state. While this method provides constant bias-variance properties at any time step, it often necessitates truncated roll-outs with shorter horizons for faster learning and policy updates within a single episode. This study highlights an unexplored issue: the bias-variance properties differ for small versus considerable time steps within truncated roll-outs. Specifically, smaller time steps may have a significant bias, prompting a need for their increase. The proposed solution involves a partial GAE update, calculating the advantage estimates for all time steps but updating the policy only for a specified range. To prevent data wastage, the data from this range is retained for further processing and policy parameter updates. This partial GAE approach, despite the increased memory requirements, promises enhanced computation speed and optimal data utilization. Empirical validation was conducted on four MuJoCo tasks and microRTS. The results show a performance improvement trend with the partial GAE estimator, outperforming regular GAE in task completion speed in microRTS. These findings offer a promising direction for improving policy update efficiency in reinforcement learning.**

I. Introduction

Reinforcement learning, a branch of machine learning that has seen numerous practical applications in recent years, revolves around an agent's interaction with its environment. The agent's approach to this interaction is guided by a systematic process: it begins by evaluating its current state based on the environment, proceeds to choose an action in response, and continues this sequence of action and evaluation, thereby perpetually learning from its environment. To formalize this process, we denote S as the set of possible states that an agent can inhabit and A as the set of potential actions an agent can take. The likelihood of an agent transitioning between states by executing a particular action is quantified by the transition probability, P.

As the agent explores through the environment, it is rewarded for its actions. These rewards, ranging from R_{min} to R_{max}, act as feedback from the environment and can be represented as a function $r : S \times A \rightarrow [R_{min}, R_{max}]$. Over time, the agent builds a trajectory sequence, τ, constituted by the states it traversed and the actions it executed, represented as $\tau : (s_1, a_1, ..., s_T, a_T)$. From this trajectory, a cumulative return, $G_t = \sum_{t=1}^{T} \gamma^{t-1} r_t$, is derived, where γ signifies the discount factor, and T denotes the total number of steps performed. The ultimate objective of reinforcement learning is to unearth an optimal policy, denoted π, which guides the agent in maximizing the expected cumulative reward.

In this context, a policy is a function that assigns a probability to each possible action in every state, formally defined as $\pi(a|s) = p(A_t = a|S_t = s)$. When operating under a particular policy, the cumulative return adheres to a specific distribution. The expected value of this cumulative return, given a starting state s, can be represented as a state-value function: $v_\pi(s) = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|S_t = s]$. Moreover, the expected cumulative return from state s after choosing a particular action a is denoted as a state-action value function: $q_\pi(s, a) = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|S_t = s, A_t = a]$.

For the purpose of estimating the value function, two methods have seen common use in reinforcement learning: Temporal Difference (TD) and Monte Carlo (MC) methods. Each method brings its own set of strengths and limitations to the table. For instance, the TD method of estimating value function is characterized by a high bias but low variance, whereas the MC method flips this trade-off, presenting low bias but high variance. To strike a balance between these two extremes, Kimura and Kobayashi (1998) [1] proposed a method termed λ return. Furthermore, the TD(λ) method, introduced by Sutton (1988)[2], serves as an extension of the λ return concept, providing a more balanced solution for value estimation.

Building on this, Schulman et al. (2015)[3] put forth the Generalized Advantage Estimation (GAE) method, which estimates the advantage value function and uses the lambda return method for the said estimation. In actual applications, as stated in the Proximal Policy Optimization (PPO) paper by Schulman et al. (2017)[4], the GAE method is truncated due to incomplete trajectories. This truncation, however, induces a notable bias in the process of estimation.

To address this shortcoming, we put forward a novel approach that we term as *partial* GAE. This technique harnesses partially computed GAEs, significantly curbing the bias induced by incomplete trajectories. In addition to carrying out experiments in widely-adopted MuJoCo environments, we test our methods in the intricate and demanding microRTS environment. The results derived from these rigorous tests demonstrate the empirical efficacy of our methods, indicating promising implications for further research and application.

II. Background

The policy gradient algorithm, which is integral to the realm of reinforcement learning, hinges on a gradient expression

*Contributed equally

typically represented as follows:

$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi t \nabla_\theta log \pi_\theta(a_t|s_t)\right] \quad (1)$$

In this formula, Ψ_t is utilized as a factor to control the magnitude of the policy update along the direction of the gradient. The rudimentary policy gradient algorithm employs the action value function, denoted as $q_\pi(s,a)$, to fill the role of Ψ_t. The action value function is typically estimated by the cumulative return, represented as G_t. Vanilla policy gradient [5] is the basic form of the policy gradient algorithm. It directly uses the full trajectory returns for gradient estimation but suffers from high variance.

Among the variety of methods devised for estimating the value function, the Monte Carlo (MC) method is arguably one of the simplest and most intuitive. This technique stems directly from the definition of the value function and utilizes the accumulated return values as the estimator for the value function. More precisely, the MC method deploys the accumulated reward, given by $\sum_{n=0}^{N} \gamma^n R_{t+n}$, from a sequence of rewards, denoted as $(r_t, r_{t+1}, ..., r_{t+N})$, as an estimate for the state value when the agent finds itself in state s_t. Owing to its comprehensive inclusion of all returns post time t, the MC method is characterized as an unbiased estimator. However, this comprehensive nature, combined with the high dimensionality, also attributes high variance to this method.

On the other hand, the Temporal Difference (TD) learning method employs $r_t + \gamma V_\theta(s_{t+1})$ as the estimator for the value function $V^\pi(s_t)$. The discrepancy between the estimated and actual value functions, referred to as e_θ, is incorporated to drive the TD algorithm, as illustrated in Equation 2. The TD method is advantageous in its comparatively low variance due to fewer random variable dimensions in the estimator. However, it introduces an estimation bias amounting to $\gamma E_{S_{t+1}}[e_\theta(S_{t+1})]$.

$$E_{(r_t, S_{t+1})}[r_t + \gamma V_\theta(S_{t+1})] = V^\pi St + \gamma E_{S_{t+1}}[e_\theta(S_{t+1})] \quad (2)$$

Striking a compromise between the bias-variance trade-off characteristic of the MC and TD methods, the λ-return method comes into play. As indicated by Equation 4, the λ-return method resorts to the TD method's estimator when λ equals 0, and to the MC method's estimator when λ equals 1.

$$G_t^{(n)} = \gamma^n V(s_{t+n}) + \sum_{l=0}^{n-1} \gamma^l r_{t+l} \quad (3)$$

$$G^\lambda t = \lambda^{N-1} G^{(N)} t + (1-\lambda) \sum_{n=1}^{N-1} \lambda^{n-1} G_t^{(n)} \quad (4)$$

To address the variance in policy gradient, the concept of a baseline, $b(t)$, is introduced into the policy gradient, as indicated in Equation 6. Utilizing the mean return from a sample as the baseline can lead to significant improvements. However, in the context of a Markov process, the baseline must be adaptive with respect to the state. Larger state baselines are appropriate when all actions have large values, and vice versa. Algorithms such as Advantage Actor-Critic (A2C) [6] and Asynchronous Advantage Actor-Critic (A3C) use V_t as the baseline. Mathematically, the advantage function $A(s,a)$ is defined as Equation 5. where $Q(s,a)$ is the action-value function and $V(s)$ is the value function.

$$A(s,a) = Q(s,a) - V(s) \quad (5)$$

The action-value function, $Q(S,A)$, can be approximated as $R + \gamma V'$. And the advantage $A(s_t, a_t)$ can be calculated as $r_t + \gamma V(s_{t+1}) - V(s_t)$.

$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} (q\pi(s,a) - b(t)) \nabla_\theta log \pi_\theta(a_t|s_t)\right] \quad (6)$$

Emphatic Temporal Difference (ETD) methods, which belong to the family of TD methods and involve the use of follow on traces, have been recognized for their theoretical prowess in resolving the deadly triad of off-policy reinforcement learning [7]. Despite the success of ETD methods, they continue to present open problems, particularly in their application to prediction and control tasks.

Truncated TD methods offer a variation on TD methods and provide a viable approach to circumventing some of the issues inherent to conventional TD methods. The Truncated Temporal Differences (TTD) technique is an approximation method that appears to alleviate the drawbacks of eligibility traces [8]. This method, despite being straightforward and requiring minimal computation per action, remains relatively unexplored [9].

In the practical realm, TD methods and their variants have found application in tasks such as mobile robot navigation. A recent study evaluated the performance of standard TD and least-squares truncated temporal-difference learning (LST2D) methods in such tasks [10].

As proposed in the Generalized Advantage Estimation (GAE) paper [3], a TD(λ)-like method calculates the estimated value as a weighted average of the estimated values of different lengths. This technique for computing the advantage is adopted by powerful reinforcement learning algorithms such as Trust Region Policy Optimization (TRPO)[11] and Proximal Policy Optimization (PPO)[4].

$$\delta_{t+l}^V = r_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l}) \quad (7)$$

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \quad (8)$$

The introduction of a discount factor, γ, to estimate the policy gradient, g^γ, results in a bias from the actual gradient, g. Such policy gradient estimation is effectively targeting the discounted cumulative reward. The concept and applications of GAE will be elaborated in subsequent sections of this paper.

There are other policy gradient algorithms besides PPO. Natural policy gradient [12] modifies the vanilla policy gradient by considering the natural metric on the space of policies. This often leads to more stable and effective learning. Deterministic policy gradients (DPG) [13] and Deep deterministic policy gradients (DDPG) [14] are policy gradient methods

designed for continuous action spaces. DDPG is the deep variant that uses neural networks for function approximation and combines ideas from DPG and actor-critic methods. Soft actor-critic [15] integrates maximum entropy reinforcement learning principles into the actor-critic framework. This promotes more exploratory policies and has shown strong performance in various tasks. Maximum a Posteriori Policy Optimisation [16] is designed to generalize well in reinforcement learning problems, especially in large-scale real-world problems and off-policy settings.

Additionally, apart from the aforementioned method for calculating Ψ_t, different strategies exist for estimating the value function. The paper by Schulman et al. [3], for instance, employs the trust region method to optimize the value function at each iteration of the batch optimization process.

$$\begin{aligned} \underset{\phi}{minimize} \quad & \textstyle\sum_{n=1}^{N} \parallel V_\phi(s_n) - \hat{V}_n \parallel \\ \text{subject to} \quad & \textstyle\frac{1}{N}\sum_{n=1}^{N} \frac{\|V_\phi(s_n) - V_{\phi_{old}}(s_n)\|}{2\sigma} \leq \epsilon \end{aligned} \tag{9}$$

Most implementations clip V_{θ_t} around the value estimates on both sides, $V_{\theta_{t-1}}$ and $V_{\theta_{t+1}}$.

$$\begin{aligned} L_t^{VF} = \max \big[& (V_{\theta_t} - V_t^{target})^2, \\ & \text{clip}\left(V_{\theta_t}, (V_{\theta_{t-1}} + \epsilon, V_{\theta_{t+1}} + \epsilon) - V_t^{target}\right)^2 \big] \end{aligned} \tag{10}$$

The study by Tucker et al. [17] proposes a technique for normalizing the advantage, demonstrating that such normalization can improve the performance of the policy gradient algorithm. Once the GAE calculates advantages in a batch, it computes the mean and standard deviation. Each advantage then undergoes a normalization process, where the mean is subtracted from it and then it is divided by the standard deviation. In Equation 11, $A^{norm}i$ represents the normalized advantage, Ai the advantage, A^{mean} the mean of the advantages, and A^{std} the standard deviation of the advantages.

$$A^{norm}i = \frac{Ai - A^{mean}}{A^{std}} \tag{11}$$

In addition, similar to GAE, V-trace [18] was a method of updating value functions that was tested in complex tasks [19]. Moreover, advantage-weighted Regression [20] fits the policy to maximize the expected advantage by using the advantage estimates as weights in a regression problem.

There are more approaches of value estimation outside the policy gradient architecture. dueling Networks [21] is introduced in the Dueling DQN architecture, this approach separates the value function into two streams: one estimating state value $V(s)$ and the other estimating the advantage function $A(s, a)$. These are later combined to produce the Q-values. Dueling architectures can lead to more stable and faster learning. Distributional value functions model [22] the entire distribution of returns, which lead to improved performance and stability in learning. Implicit Quantile Networks (IQN) [23] is an extension of distributional value functions. Instead of modeling the return distribution with fixed quantiles, IQN samples random quantiles, providing a more flexible approximation of the return distribution.

Overall, the estimation of a more instructive value function is of great significance for the performance of the policy gradient algorithm. In the subsequent parts of this paper, the practical application of GAE will be explored in depth, alongside potential ways to improve its functionality. This includes an analysis of the function approximations used, the impact of varying hyperparameters, and strategies for further enhancement of policy gradient performance.

III. PARTIAL GAE

The bias-variance trade-off has always been a significant challenge in machine learning. In the estimation of the value of RL, as mentioned above, achieving satisfactory levels of bias and variance simultaneously often proves difficult. In practical applications, a fixed length is typically assigned to each sample to enhance the efficiency of parallel computing. This approach is also adopted when sampling a complete trajectory becomes too time-consuming. Instead, only a portion of the complete trajectory is sampled at a time to expedite the training process. As discussed in the PPO paper [4], a truncated Generalized Advantage Estimation (GAE) is used for fixed-length trajectory segments (Figure 1). The GAE of an incomplete trajectory with length T is represented as follows:

$$\hat{A}_T^{GAE(\gamma,\lambda,T)} = \delta_T^V = r_T + \gamma \cdot 0 - V(s_T) \tag{12}$$

$$\hat{A}_t^{GAE(\gamma,\lambda,T)} = \sum_{l=0}^{T-t} (\gamma\lambda)^l \delta_{t+l}^V \tag{13}$$

In the GAE paper [3], the sum of k δ terms is denoted as $\hat{A}_t^{(k)}$. When $k = 1$, $\hat{A}_t^{(1)} = \delta_t^V$, which exhibits a large bias but low variance. However, as k increases, the bias progressively decreases.

$$\hat{A}_t^{(k)} = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + \gamma^k V(s_{k+t}) + \sum_{l=0}^{k-1} \gamma^l r_{t+l} \tag{14}$$

The GAE, $\hat{A}^{GAE(\gamma,\lambda,T)}$, is essentially the sum of exponentially-weighted $\hat{A}_t^{(k)}$ terms. As $T - t$ increases and the number of cumulative terms grows, the bias is reduced.

$$\hat{A}_t^{GAE(\gamma,\lambda,T)} = (1-\lambda) \sum_{k=1}^{T-t+1} \lambda^{k-1} \hat{A}_t^{(k)} \tag{15}$$

For the special cases where $T = D$ and $t = T$, with D being the end of the complete trajectory, it is crucial to note that when $T = D$, $\hat{A}_t^{GAE(\gamma,\lambda,T)}$ is identical to $\hat{A}_t^{GAE(\gamma,\lambda,D)}$, and its bias is minimized.

$$\hat{A}_t^{GAE(\gamma,\lambda,D)} = (1-\lambda) \sum_{k=1}^{D-t+1} \lambda^{k-1} \hat{A}_t^{(k)} \tag{16}$$

$$\hat{A}_T^{GAE(\gamma,\lambda,T)} = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{17}$$

Like the infinite $\hat{A}^{GAE(\gamma,\lambda)}$, the truncated GAE strives to balance bias and variance via λ. While $\hat{A}^{GAE(\gamma,1,T)}$ exhibits low bias and high variance, $\hat{A}^{GAE(\gamma,0,T)}$ shows low variance but high bias. Nevertheless, the truncated GAE generally

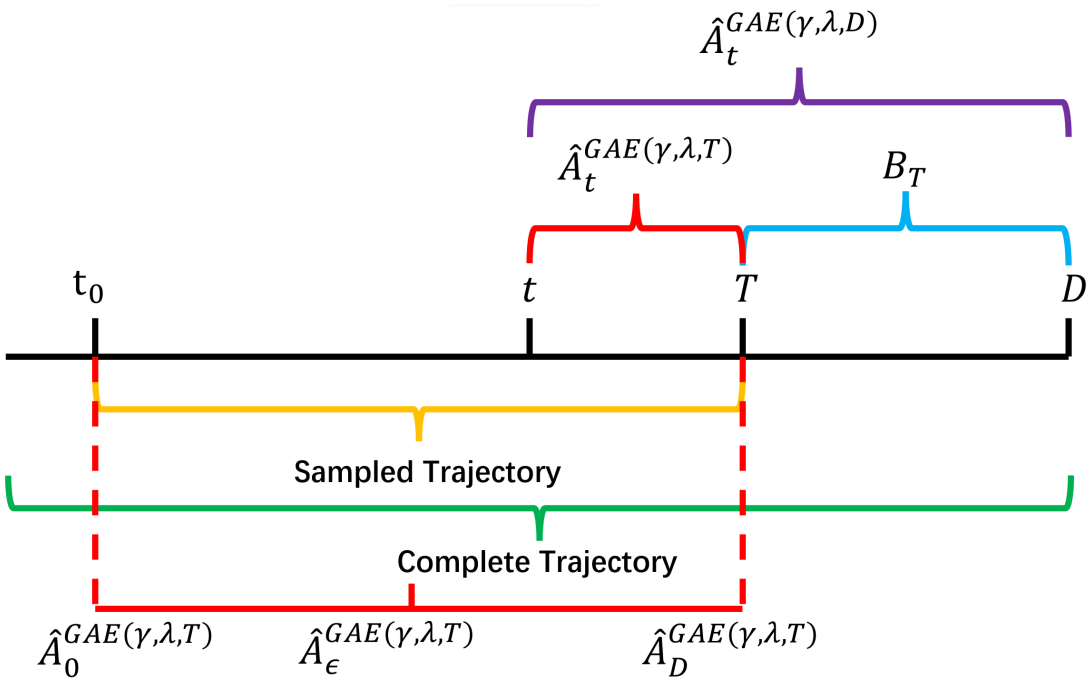

Fig. 1. The black line in the figure represents the complete trajectory of an environment from start to finish. In practical applications, to accommodate parallel computing and prevent excessively long episodes, a fixed sampling length is adopted. Since the last step of the sampled trajectory does not coincide with the termination of the full trajectory, a truncated GAE is used as the estimator. The red part illustrates the trajectory obtained through actual sampling and the truncated GAE calculation based on this trajectory. The purple part depicts the GAE that would be calculated if the trajectory were complete. The blue part indicates the bias resulting from the exclusion of unsampled portions of the trajectory from the GAE calculation. For all of the calculated truncated GAE $\hat{A}_t^{GAE(\gamma,\lambda,T)}$, we propose to use $\hat{A}_t^{GAE(\gamma,\lambda,T)}$ for updating in the case $t \leq \epsilon$ and discard $\hat{A}_t^{GAE(\gamma,\lambda,T)}$ when $t > \epsilon$ as it shown in red part of this figure.

presents a higher bias and lower variance compared to its infinite counterpart. From a broader perspective, minimizing bias is crucial as it makes $\hat{A}_t^{GAE(\gamma,\lambda,T)}$ increasingly informative for a specific trajectory as the step t is reduced.

The difference between the truncated $\hat{A}_t^{GAE(\gamma,\lambda,T)}$ and the GAE of a complete trajectory $\hat{A}_t^{GAE(\gamma,\lambda,D)}$ is denoted as B_t.

$$B_t = \hat{A}_t^{GAE(\gamma,\lambda,D)} - \hat{A}_t^{GAE(\gamma,\lambda,T)} = \sum_{l=T-t}^{D-t} (\gamma\lambda)^l \delta_{t+l}^V \quad (18)$$

According to the Equation 19, B_t decreases exponentially with the step t for a specific trajectory length T, given that $0 < \gamma < 1$. This decrease in B_t diminishes the deviation between $\hat{A}_t^{GAE(\gamma,\lambda,T)}$ and $\hat{A}_t^{GAE(\gamma,\lambda,D)}$. Since $\hat{A}_t^{GAE(\gamma,\lambda,T)}$ possesses a larger bias than $\hat{A}_t^{GAE(\gamma,\lambda,D)}$, the reduction of B_t helps to decrease the bias of $\hat{A}_t^{GAE(\gamma,\lambda,T)}$.

$$B_t = \sum_{l=0}^{D-T} (\gamma\lambda)^{l+T-t} \delta_{T+l}^V = (\gamma\lambda)^{T-t} B_T \quad (19)$$

At the same time, it should be noted that when $(T-t)$ increases, the variance of $\hat{A}_t^{GAE(\gamma,\lambda,T)}$ will increase. A brief mathematical explanation is given below.

If $\mathrm{Var}(\hat{A}_{t-1}^{GAE(\gamma,\lambda,T)}) > \mathrm{Var}(\hat{A}_t^{GAE(\gamma,\lambda,T)})$ holds, then the proposition is true. The variance of the sum $D_1 + D_2$ is given by Equation 20.

$$\mathrm{Var}(D_1+D_2) = \mathrm{Var}(D_1)+\mathrm{Var}(D_2)+2\times\mathrm{Cov}(D_1,D_2) \quad (20)$$

For the variance of $D_1 + D_2$ to be greater than the variance of D_1, the following must be true:

$$\mathrm{Var}(D_2) + 2 \times \mathrm{Cov}(D_1, D_2) > 0 \quad (21)$$

From the Inequality 21, if $\mathrm{Var}(D_2) > 0$ (which is always true for any non-constant distribution) and the covariance term $\mathrm{Cov}(D_1, D_2)$ is positive, then the inequality holds. Thus, in our context, it is only necessary to prove that the covariance is greater than 0.

The covariance between two random variables D_1 and D_2 is a measure of how the two variables change together. Mathematically, covariance is defined as:

$$\mathrm{Cov}(D_1, D_2) = E[(D_1 - E[D_1])(D_2 - E[D_2])] \quad (22)$$

Where $E[\cdot]$ denotes the expected value. When D_1 tends to be above its mean value at the same time that D_2 is above its mean value, or similarly for being below their mean values. In simpler terms, if D_1 increases as D_2 increases (on average), or D_1 decreases as D_2 decreases, then their covariance is positive.

From Equation 7, δ comes from the same distribution related to $V(s)$ for different t. $\sum_{l=0}^{T-t}(\gamma\lambda)^l\delta^V$ changes in the same direction as $(\gamma\lambda)^{T-t+1}\delta^V$. This means that the covariance of the two is greater than 0, as shown in Equation 23.

$$\mathrm{Cov}(\sum_{l=0}^{T-t}(\gamma\lambda)^l\delta^V, (\gamma\lambda)^{T-t+1}\delta^V) > 0 \quad (23)$$

Algorithm 1 PPO with partial GAE

```
for iteration=1, 2, . . . do
  for actor=1, 2, . . . , N do
    Run policy π_{θ_old} in environment, get samples (s_1, a_1, ..., s_T, a_T)
    Compute advantage estimates Â_1, ..., Â_T
    if T is not done then
      only use Â_1, ..., Â_ϵ for updating
      keep (s_ϵ, a_ϵ, ..., s_T, a_T) as (s_1, a_1, ..., s_{T−ϵ}, a_{T−ϵ})
    else if T is done then
      use Â_1, ..., Â_T for updating
    end if
  end for
  for epoch K do
    Optimize surrogate L wrt θ, with minibatch size
    θ ← θ_old
  end for
end for
```

From the equation 13 and inequality 23, it can be get:

$$\text{Var}(\sum_{l=0}^{T-t}(\gamma\lambda)^l\delta^V + (\gamma\lambda)^{T-t+1}\delta^V) > \text{Var}(\sum_{l=0}^{T-t}(\gamma\lambda)^l\delta^V) \quad (24)$$

Therefore $\text{Var}(\hat{A}_{t-1}^{GAE(\gamma,\lambda,T)}) > \text{Var}(\hat{A}_t^{GAE(\gamma,\lambda,T)})$, that is, as (T-t) increases, the variance of truncated GAE will increase.

In conclusion, the practice of using fixed-length sampling trajectories in practical applications results in truncated GAE calculations. This truncation leads to significant bias, particularly when t approaches the end of the trajectory. A large bias can cause underfitting, where the learning algorithm consistently underestimates the value of certain states or actions. This can skew the learning process, leading the agent to undervalue certain states or actions, potentially resulting in sub-optimal behavior. To address this, we suggest adopting a "partial GAE" approach. Essentially, we propose using a section of the GAE and discarding the remaining part of the trajectory, typically characterized by substantial bias.

We thus propose a Proximal Policy Optimization algorithm employing partial GAE, as outlined in Algorithm 1. This method involves a partial parameter, ϵ, and a sample length, T. During each iteration, each sampler collects T samples and computes T truncated GAEs. When $t > \epsilon$, we consider a portion of a GAE; thus, the advantage estimates at time t in the trajectory are excluded from the training.

Following this, we construct the surrogate loss as defined in [4], based on these $N(T-\epsilon)$ data. Then, we optimize the policy through minibatch Adam optimization for K epochs. For Algorithm 1, it is important to note that as $T-t$ increases, the bias of $\hat{A}_t^{GAE(\gamma,\lambda,T)}$ decreases. Also, as the partial parameter decreases, the bias of $\hat{A}_1, ..., \hat{A}_\epsilon$ reduces as well. This dynamic ultimately leads to a more effective learning process, reinforcing the practical utility of our proposed partial GAE method.

IV. EXPERIMENTS

Our methodology was examined and put through a rigorous evaluation across two distinct environments: MuJoCo [24] and microRTS [25]. MuJoCo, an acronym for Multi-Joint dynamics with Contact, is a physics engine or, more precisely, a physical simulation platform. It offered us the opportunity to conduct primary experiments in the Ant-v3 environment, a robotics task involving a simulated quadruped robot. Our analysis was further supported by supplemental tests in several other environments like Halfcheetah-v3, Hopper-v3, and Walker2d-v3, each with unique characteristics.

In contrast to MuJoCo's continuous and physically simulated environment, we also engaged with microRTS. This streamlined and highly abstracted real-time strategy (RTS) game environment is essentially a grid-based environment with RTS game features. Unlike MuJoCo, it offers discrete states and actions, and a significantly large number of game-play steps, creating a vastly different experimental backdrop.

For the MuJoCo experiments, we opted to use Version 1.31, distributed under an open-source MIT License. To achieve optimal results, we carefully configured the hyperparameters according to industry standard practices. This included setting a discount factor (γ) of 0.99, a Generalized Advantage Estimation (λ for GAE) of 0.95, a Proximal Policy Optimization (PPO) clip coefficient of 0.2, and a value coefficient of 1. Moreover, the optimizer's learning rate was set to 2.5e-4, across 32 environments, over a period of 2 epochs. In the architecture of the policy representation, we implemented a fully connected three-layer Multi-layer Perceptron (MLP) with a 64-unit hidden layer. To further enhance the model's learning capability, an additional two noise layers were appended after the MLP to facilitate exploration during the training process. In the Ant-v3 games, we conducted games consisting of 1000 steps, with experiments for a sample length (T) ranging from 128 to 1024 and parameter (ϵ) from 64 to 512.

In our journey of testing the methodology in the microRTS environment, we conducted experiments against CoacAI, an AI that emerged victorious in the 2020 microRTS competition. We maintained consistency by employing the same values for the discount factor (γ), Generalized Advantage Estimation (λ for the GAE), PPO clip coefficient, value coefficient, optimizer's learning rate, number of environments, and epochs

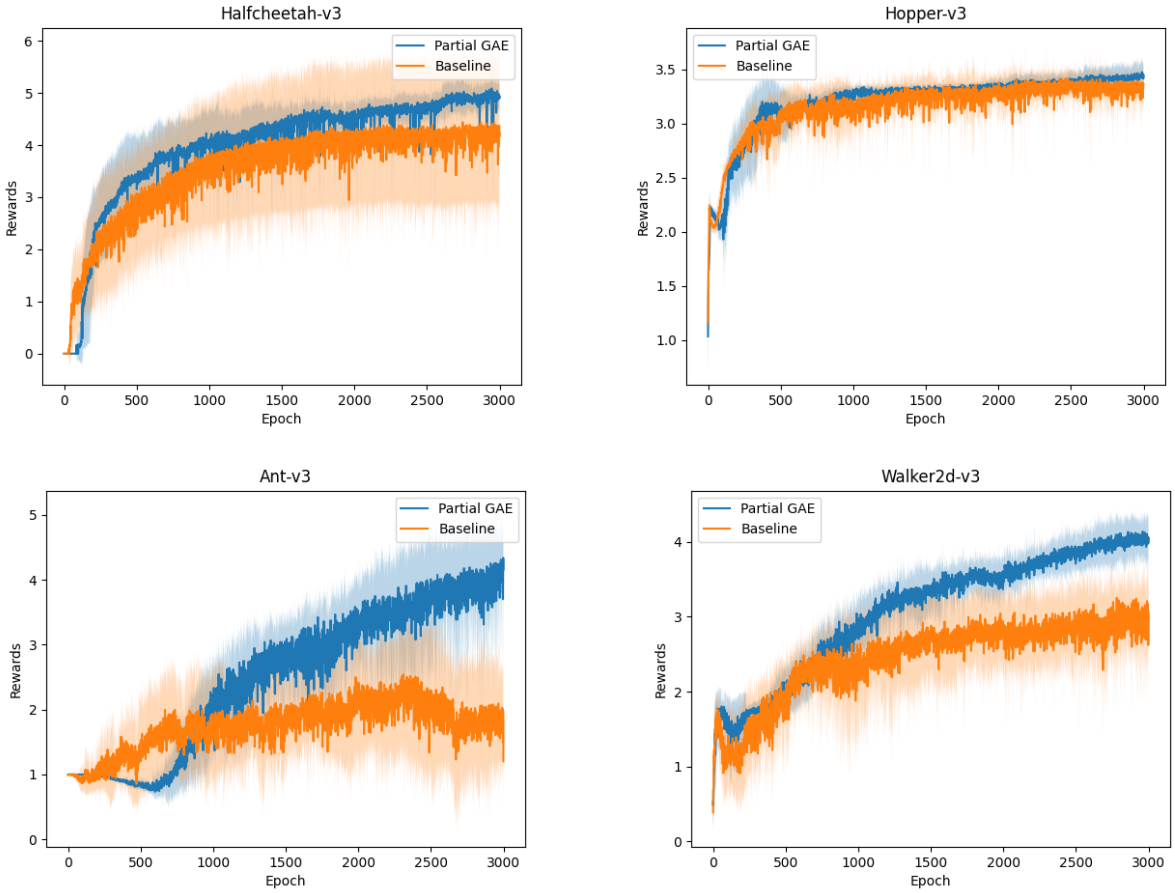

Fig. 2. Training curve in different MuJoCo environments. In the experiment, each epoch contains 8192 steps. The baseline in the experiment is common PPO, and the partial parameter of the partial GAE group is 256.

as those we used in the MuJoCo experiments. The one and only variance from the MuJoCo setup was the introduction of an entropy regularization coefficient of 0.01. When it came to policy representation, we used a two-layer convolutional neural network attached to a fully connected three-layer MLP, with a significantly larger 512-unit hidden layer.

To assess the efficiency and effectiveness of our methodology, we relied heavily on results obtained under specific sampling conditions. In the context of the MuJoCo environment, we employed the total reward per episode during training as the performance score. In the microRTS setting, the evaluation criterion was the win rate of the most recent 100 episodes following a predefined training period. We ensured consistency and randomness by running each set of variables with 10 random seeds. All experiments were conducted on an Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz.

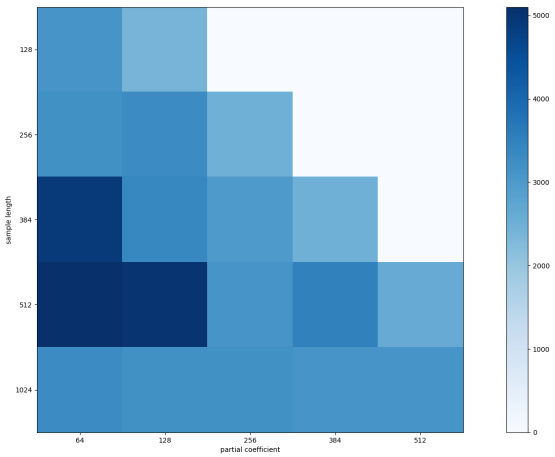

Fig. 3. The performance after training in Ant-v3. Since sample time T is greater than or equal to partial parameter ϵ, the white part has no data. $T = \epsilon$ means not to discard any GAE, which is the baseline.

A. Experiments in MuJoCo

As was expounded upon in the previous sections, the value of t in the truncated Generalized Advantage Estimation (GAE) $\hat{A}^{GAE(\gamma,\lambda,T)}t$ plays a crucial role. Specifically, both a smaller t value and a larger sample length T tend to result in a decrease in the bias of the estimate. Conversely, it also leads to an increase in the variance. And the variance of $\hat{A}_t^{GAE(\gamma,\lambda,T)}$ consistently remains smaller than that of $\hat{A}_t^{GAE(\gamma,\lambda,D)}$.

Theoretically, the partial parameter ϵ should be as small as feasible to decrease the bias and sample length T should be

as large as feasible. However, this theoretical standpoint faces challenges when applied in practical scenarios. In real-world applications, employing small partial parameters can lead to an increase in the number of GAE calculations required. This might potentially escalate the complexity of computational operations. On the other hand, opting for a larger sampling length results in a longer sequence that needs to be processed in a single GAE calculation, which in turn increases the calculation time. At the same time, the increase in variance should be taken into account, which further highlights the need for a trade-off.

To further scrutinize these dynamics, we executed a comparison of the common Proximal Policy Optimization (PPO) as the baseline with the partial GAE PPO across several environments: Ant-v3, Halfcheetah-v3, Hopper-v3, and Walker2d-v3. In addition, we compared the effects of different partial parameters and sample lengths specifically in the Ant-v3 environment. In an attempt to ensure a clean and undisturbed comparison, we excluded interference from additional methods. Therefore, our implementation did not incorporate any of the tricks outlined by Engstrom et al. [26], such as value clipping and an advantage normal.

The fruits of our experimental labor are illustrated in Figure 2. In this figure, one can observe that utilizing the partial GAE yields better performance than the baseline. Furthermore, a noteworthy trend emerges wherein an improvement in performance is achieved by employing smaller partial parameters. The corresponding heat-map in Figure 3 showcases the performance after one hour of training as T and ϵ are varied. A meticulous inspection reveals that the peak performance score coincides with the partial parameter ϵ lying within the range of [384, 512] and the sample length T falling in the range of [64, 128], specifically in the Ant-v3 environment.

It's important to highlight that performance improvements do not occur indefinitely by continuing to increase the partial parameter or decrease the sampling length, especially when the partial parameter is small or the sampling length is large. This phenomenon can be attributed to the fact that, as evidenced by Equation 19, when $(T - t)$ becomes significantly large, B_t tends to shrink considerably, and the alteration brought about by persistently increasing $(T - t)$ becomes negligible. According to the worker of PPO paper [4] and GAE paper [3] while GAE indeed does an excellent job of drastically reducing variance by sacrificing bias, and the variance of the truncated GAE is lower than that of the complete trajectory, based on our experiments' results this doesn't imply that the sole aim should be to minimize bias while disregarding the impact of variance. In the realm of practical applications, it's essential to find the median of the partial parameter ϵ and sample length T to strike a balance between bias and variance. This essentially implies that there exists an intermediate value that optimizes the training effect, rather than just focusing on maximizing $(T - \epsilon)$.

B. Experiments in microRTS

To further validate and enrich our research findings, we ventured into additional experimentation within the microRTS environment. MicroRTS is known for being a sparsely rewarded setting that presents an array of extensive game steps, adding complexity and depth to the conducted experiments. The specific tasks in this environment require participating agents to acquire and demonstrate proficiency in key mechanics of real-time strategy (RTS) games. These mechanics are multifaceted, including of resource collection, construction of game units, and the strategic elimination of enemy units and their bases. For these supplemental experiments, we set the sample length T to 512.

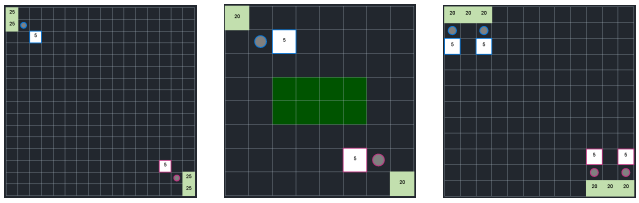

Fig. 4. Initial states of maps for experiments. From left to right maps are "basesWorkers16x16", "basesWorkers8x8Obstacle", "TwoBasesWorkers12x12"

In our endeavor to create a comprehensive testing scenario, we selected three distinct maps: "basesWorkers16x16", "basesWorkers8x8Obstacle", and "TwoBasesWorkers12x12". This diverse selection of maps aimed to encapsulate various map sizes and initial conditions, thereby ensuring that the experimental results weren't biased towards any particular configuration.

It's worth noting that due to its extensively long game steps, the microRTS environment inherently introduces a larger bias into the value estimate when truncated GAE is used. As highlighted in previous sections, substantial bias in value estimation when employing Proximal Policy Optimization (PPO) can have several negative ramifications. One key consequence is that it can significantly undermine learning efficiency, primarily because inaccurate estimations fail to reflect the expected return of a state or state-action pair accurately. As a result, the agent ends up learning sub-optimal policies. Secondly, a substantial level of bias can disrupt the delicate balance between exploration and exploitation for the agent, potentially leading to sub-optimal policy decisions as the agent might consistently undervalue or overvalue specific actions. Another repercussion is the potential for an increase in training instability, triggered by inaccurate advantage estimates originating from a biased value function, leading to unpredictable and inefficient policy updates. Lastly, the agent might encounter difficulties in converging to the optimal policy, given that flawed feedback inaccurately guides policy updates.

These issues assume heightened significance in the context of our microRTS experiments. The Figure 5 reveals a telling story about the performance on the three different map experiments. Initial states of maps for these experiments are shown in Figure 4. Initially, the baseline method displays a learning curve strikingly similar to the method using partial GAE. However, as training progresses, the baseline method struggles to keep up, faltering in its quest to achieve a higher win rate and reward. In contrast, the method utilizing partial GAE demonstrates significant improvement, illustrating the

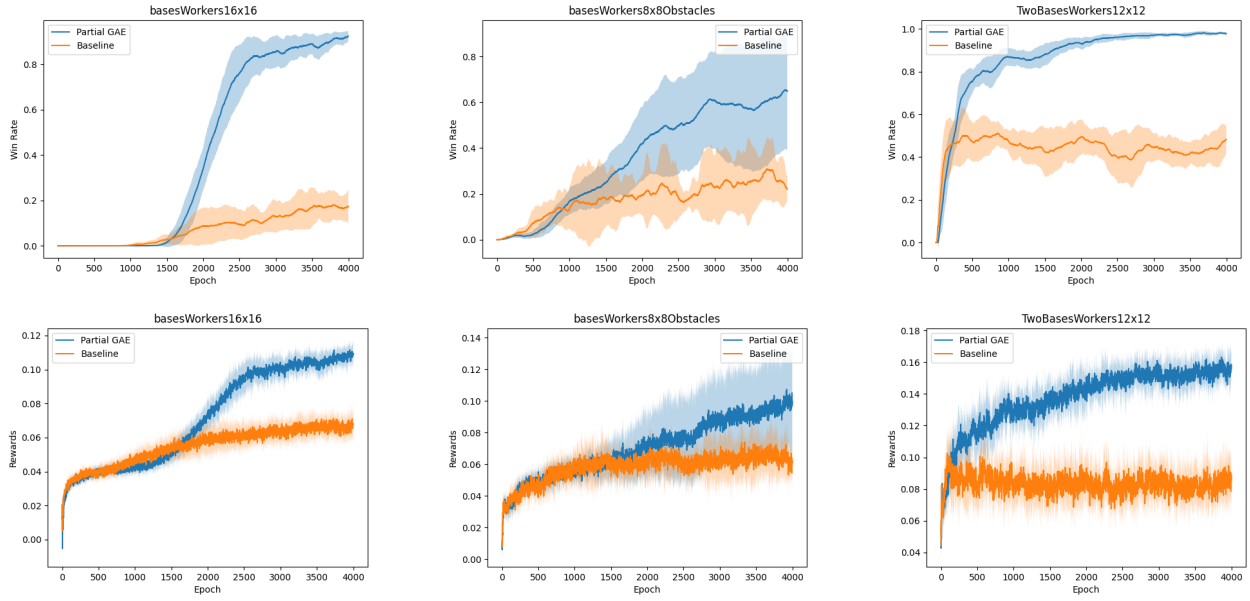

Fig. 5. Learning curve of win rate and rewards in different maps. In the experiment, each epoch contains 8192 steps. Each column represents a different map, the first row is the winning rate in different maps, and the second row is the rewards in different maps. The baseline in the experiment is common PPO, and the partial coefficien of the partial GAE group is 256.

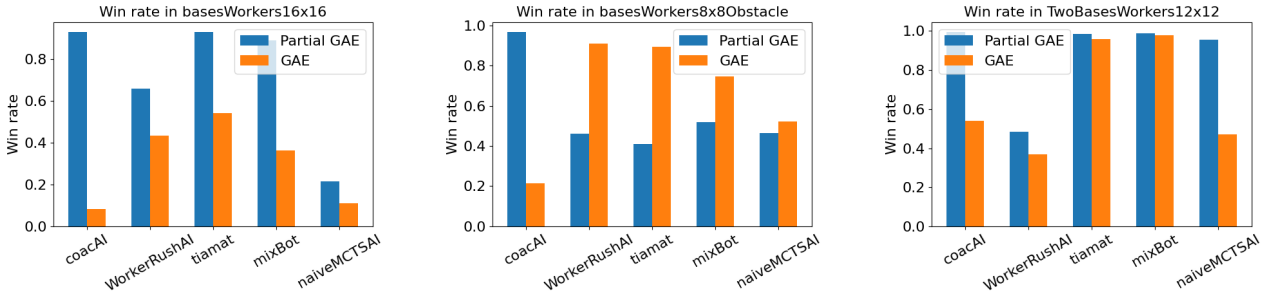

Fig. 6. Trained model against different AI in 1000 games, where CoacAI is 1st place in 2020, tiamat is 1st place in 2018, MixedBot is 2nd place in 2019, WorkerRushAI is 8th place in 2020, NaiveMCTS is 9th place in 2020.

benefits of this approach by eventually converging to a win rate of nearly 100%.

The comparative performance of our trained model against other AI counterparts is depicted in Figure 6. It is evident from the figure that our PAE-trained model exhibits superior performance on larger 12x12 and 16x16 maps. On the other hand, the model utilizing GAE shows better performance on smaller 8x8 maps. This difference in performance outcomes invites an investigation into the underlying reasons. A conjecture we put forward is that the PAE model could be overfitting when competing against coacAI on the smaller 8x8 map. Overfitting signifies a situation where the model is excessively adapted to the training data, impairing its ability to generalize to unseen data. This event might be more likely to transpire on smaller maps due to the smaller state space and the fewer variables to account for. Moreover, double-descent risk curve [27] as mentioned by Belkin et al. Smaller neural networks are more susceptible to overfitting, whereas larger, more complex networks are less so. Given that the complexity of the neural network correlates with the map size in our experiments, methods which learn faster on smaller maps may be more inclined towards overfitting. Consequently, our PAE model could be over-learning the specific patterns of coacAI, which adversely impacts its versatility and effectiveness in unfamiliar situations. Nonetheless, it's critical to recognize that across all map sizes, our PAE model consistently exhibited superior performance in overcoming the specific adversary it was trained against. This implies that the PAE model has a strong capability to learn and optimize its performance based on the traits of its opponent.

Additionally, we examine different parameters on maps of different sizes (but similar initial states as shown in Figure 8). As shown in Figure 7. As explained in Section 3, the most effective partial parameter is an intermediate value that requires a trade-off between bias and variance. However, it can also be seen from the figure that the results of using partial GAE are often better than not using it.

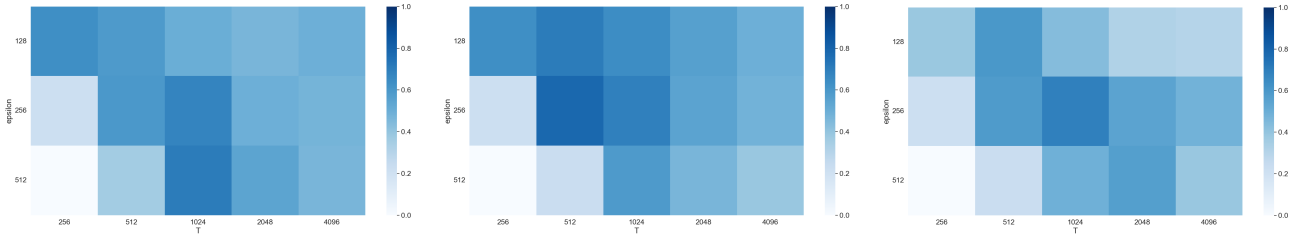

Fig. 7. Winning rate with different sample time T and partial parameter ϵ after training in different map. $T = \epsilon$ means not using partial GAE, which is the baseline. From left to right "basesWorkers10x10","basesWorkers16x16", "basesWorkers24x24"

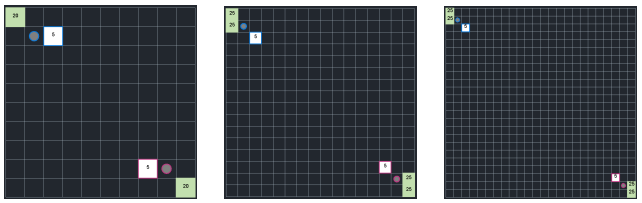

Fig. 8. Experiments in different size maps with similar initial states. From left to right maps are "basesWorkers10x10", "basesWorkers16x16", "basesWorkers24x24"

V. CONCLUSION

In conclusion, this paper introduces and explores a novel concept of Partial Generalized Advantage Estimation (Partial GAE) in the context of Reinforcement Learning (RL), focusing particularly on Proximal Policy Optimization (PPO). We analyzed the bias-variance trade-off that arises in traditional GAE when considering trajectories of fixed lengths for more efficient parallel computing and sampled only a part of a complete trajectory for more effective training. Theoretical underpinnings were provided, showing how Partial GAE can reduce bias while maintaining computational efficiency.

We put forward a variant of the PPO algorithm with Partial GAE, with a particular emphasis on the role of the partial parameter in reducing bias. This new approach offers a systematic way to decrease the bias of the GAE estimator by leveraging trajectory information effectively.

The paper also presents experimental results that demonstrate the effectiveness of the proposed Partial GAE in both continuous control tasks using MuJoCo and discrete control tasks in the microRTS environment. Across these varied experimental landscapes, the proposed method, in most cases, surpassed the conventional approach, affirming the theoretical propositions underpinning this study.

REFERENCES

[1] H. Kimura, S. Kobayashi *et al.*, "An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value function." in *ICML*, vol. 98, 1998.

[2] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[3] 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.

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

[5] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[6] 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*. PMLR, 2016, pp. 1928–1937.

[7] S. Zhang and S. Whiteson, "Truncated emphatic temporal difference methods for prediction and control," *The Journal of Machine Learning Research*, vol. 23, no. 1, pp. 6859–6917, 2022.

[8] P. Cichosz and J. J. Mulawka, "Fast and efficient reinforcement learning with truncated temporal differences," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 99–107.

[9] P. Cichosz, "Truncating temporal differences: On the efficient implementation of td (lambda) for reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 2, pp. 287–318, 1994.

[10] J. Ren, Y. Lan, X. Xu, Y. Zhang, Q. Fang, and Y. Zeng, "Deep reinforcement learning using least-squares truncated temporal-difference," *CAAI Transactions on Intelligence Technology*, 2023.

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

[12] S. M. Kakade, "A natural policy gradient," *Advances in neural information processing systems*, vol. 14, 2001.

[13] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. Pmlr, 2014, pp. 387–395.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[16] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller, "Maximum a posteriori policy optimisation," *arXiv preprint arXiv:1806.06920*, 2018.

[17] G. Tucker, S. Bhupatiraju, S. Gu, R. Turner, Z. Ghahramani, and S. Levine, "The mirage of action-dependent baselines in reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5015–5024. [Online]. Available: https://proceedings.mlr.press/v80/tucker18a.html

[18] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *International conference on machine learning*. PMLR, 2018, pp. 1407–1416.

[19] 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.

[20] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *arXiv preprint arXiv:1910.00177*, 2019.

[21] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[22] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 449–458.
[23] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 1096–1105.
[24] OpenAI, "Gym-mujoco," https://www.gymlibrary.ml/environments/mujoco/, 2022, accessed: 2022.
[25] S. O. Villar, "microrts," https://github.com/santiontanon/microrts, 2017, accessed: 2017-07-13.
[26] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation matters in deep policy gradients: A case study on PPO and TRPO," *International Conference on Learning Representations*, 2020.
[27] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, 2019.