Vol.2022-GI-47 No.18
2022/3/19

スパコンにおける PPO の性能評価

桑村佳佑[1]　大島聡史[2]　片桐孝洋[2]　永井亨[2]

概要：機械学習手法の一つである強化学習を実空間やビデオゲームなどに用いる研究が注目を集めている．強化学習ではエージェントがいかに効率的に経験サンプルを集められるかが重要となる．本稿では強化学習アルゴリズムの評価のためのシミュレーション環境の一つである Slime Volleyball Gym Environment を強化学習アルゴリズム PPO によって学習し，得られたモデルの性能評価を行った．実験環境として名古屋大学情報基盤センターに設置されたスーパーコンピュータ「不老」を使用し，モデルの性能と学習時間の関係についても評価を行った．

キーワード：強化学習，PPO，スパコン

1. 背景

近年，ゲーム AI の分野では強化学習をビデオゲームなどの環境に用いる研究が盛んにおこなわれている．強化学習とは，ある環境内でエージェントが得る報酬の総和を最大化するような意思決定則(方策)を学習する手法である．強化学習に深層学習の技術を取り入れた深層強化学習の登場により飛躍的に性能が向上し，様々なゲームのタスクにおいて人間のスコアを上回るほどの AI の学習に成功している．

強化学習は，教師あり学習と異なり教師データが存在しない．そのため，エージェントは環境との相互作用によって得られえた経験データをもとに学習を行う．強化学習では，いかに効率的に経験データを集められるかが重要となる．近年の強化学習アルゴリズムには分散強化学習を用いるものが多く，エージェントの並列数を増やすことで経験データを効率的に収集している．

本研究ではスーパーコンピュータ（スパコン）を使い，エージェントの並列数を増やすことで,並列数と学習時間，スコアの関係について性能評価を行った．

2. 強化学習

本章では強化学習の概要，特に本研究で用いた強化学習アルゴリズムにつながる前提知識について述べる．

強化学習とは，機械学習の一種で特に方策を学習する手法のことを指す．強化学習の目的は，ある意思決定問題を扱う環境内（ビデオゲームなど）において，エージェントが得る報酬の総和を最大化する方策を得ることである．エージェントの行動（意思決定）は，方策に従って行われる．方策は$\pi(a|s)$であらわされ，これをある環境の状態sにおいて，ある行動aをとる確率とする．行動が環境に入力されると，報酬rと次の状態が環境から出力される．エージェントは各時刻tにおいて，状態をもとに方策に従って行動を選択することを繰り返す．環境から得る報酬の総和の期待値は$\mathbb{E}_\pi[\sum_{t=0}^{T} r_t]$であらわされる．強化学習ではこれを最大化する方策を見つけることが目標となり，様々な手法が提案されている．

3. Proximal Policy Optimization Algorithms

Proximal Policy Optimization Algorithms（PPO）[1]は 2017 年に John Schulman らによって提案された強化学習アルゴリズムである．PPO は方策勾配法の一種と位置付けられ，またそれ以外にも Actor-Critic や分散強化学習などの手法が内包されている．本章では 1 節から 3 節で，従来の強化学習アルゴリズムで提案された手法を，4 節では PPO で新たに提案された Clipped Surrogate Objective をそれぞれ解説する．なお一般に PPO の実装といえば，この Clipped Surrogate Objective の実装を指すことが多い．

3.1 方策勾配法

方策勾配法では，環境から得られる報酬の総和の期待値（期待収益$J(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{T} r_t]$）を最大化するために，パラメータ$\theta$を用いて方策$\pi_\theta$を直接改善する手法をとる．$\theta$は確率的勾配降下法によって以下のように更新できる．

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta) \tag{1}$$

ここでαは学習率，$J(\theta)$は最大化したい目的関数である．更新には$J(\theta)$の微分値が必要となるが，これについて式(2)の方策勾配定理が成り立つことが知られている．

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, Q^{\pi_\theta}(s,a)] \tag{2}$$

ここで$Q^{\pi_\theta}(s,a)$は行動価値関数と呼ばれる．これは状態sで行動aを選択したあと，方策π_θに従い続けた場合に得る期待収益のことで，Q 関数とも呼ばれる．

3.2 Actor-Critic

Actor-Critic とは方策の改善モデル Actor と，その方策の価値を評価するモデル Critic を同時に学習させる手法のことを指す．方策勾配定理に含まれている行動価値関数は，式(1)でθが更新される度に変化するため，これを直接用い

1 名古屋大学 大学院情報学研究科
Graduate School of Informatics, Nagoya University
2 名古屋大学 情報基盤センター
Information Technology Center, Nagoya University

ることが難しいという問題がある．そこで Critic によって行動価値関数を関数近似器で置き換えることで解決を図っている．

なお PPO の Actor-Critic では行動価値関数の代わりに状態価値関数$V^{\pi_\theta}(s)$を近似する．これはある状態sにおいて方策π_θに従い続けた場合に得る期待収益のことを指す．つまり状態価値関数は行動に依存しない価値関数である

3.3 分散強化学習

PPO で使われている分散強化学習は A3C[2]で提案された手法を用いている．これは複数のエージェントがそれぞれの環境で並列に動作し経験を集める手法のことを指す．分散強化学習の目的は，経験同士の時系列的な相関の低減である．確率的勾配降下法ではサンプル間の独立性を前提としている．そのため経験同士に時系列的な相関があると局所解に陥りやすいという問題がある．複数エージェントから集めた経験を学習に用いることで，全体としての経験同士の時系列的な相関が減り，学習が安定化する．A3C では分散強化学習により，経験の収集にかかる時間の短縮やモデルの性能向上が報告されている．

3.4 Clipped Surrogate Objective

PPO では方策関数が大きく更新されすぎることを防ぐために，Clipped Surrogate Objective と呼ばれる代理目的関数を提案している．まず，パラメータの更新前後の方策関数の出力の比を$r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$と表す．ここでの$r$は報酬の$r$ではないことに注意されたい．これを用いた代理目的関数$L^{CLIP}(\theta)$を式(3)に示す．

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \quad (3)$$

なおϵはハイパーパラメータであり，A_tはアドバンテージ関数$A_t = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$と呼ばれる．式中に用いられている clip 関数の定義を式(4)に示す．

$$\mathrm{clip}(r,x,y) = \begin{cases} x, & r < x \\ y, & r > y \\ r, & x \le r \le y \end{cases} \quad (4)$$

clip 関数によって，更新前後の方策の比を一定の範囲でクリップする．さらにクリップする前後を比較し小さいほうを使用する．PPO ではこの工夫によって，パラメータの急激な更新を抑え，学習を安定させる．

4. Slime Volleyball Gym Environment

4.1 概要

Slime Volleyball Gym Environment[3]とは，強化学習アルゴリズムの性能評価を行うためのシミュレーション環境である．同様の環境として OpenAI が提供する OpenAI Gym[4]の Atari ゲーム環境がある．OpenAI Gym には，OpenAI が用意する環境以外にサードパーティーからも環境が提供されており，Slime Volleyball Gym Environment もそのうちの一つである．

4.2 ルール

本節では Slime Volleyball Gym Environment のルールについて解説する．このゲームは左右 2 体のスライムがバレーボールを行う 2D アクションゲームである．各スライムは残機を 5 つ持つ．勝利条件は先に残機が 0 になった方が負け，もしくは制限時間 1 分経過時に残機の多い方が勝ちの 2 つである．ゲームプレイは以下のような流れで行われる．

(1) 初期状態（図 1）
(2) 中央のボールがランダムに動き出し，以降自由落下
(3) スライムは左右移動とジャンプによって，ボールを地面に落とさないよう跳ね返す
(4) ボールが地面に落ちた時，落としたスライムの残機が 1 つ減る
(5) ボールのみが初期状態の位置に戻り，(2)に戻る

なお，実際のバレーボールと異なり，一度のラリーで何度ボールに触れてもよい．また，中央の柵を飛び越えることはできない．

スコアは自分のコートにボールが落ちた時に-1，相手のコートにボールが落ちた時に+1 が与えられえる．

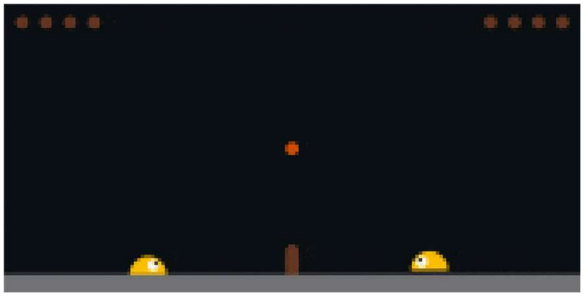

図 1 ゲーム画面（初期状態）

4.3 状態空間と行動空間

Slime Volleyball Gym Environment には状態空間と行動空間の異なる 3 つの環境が用意されている．

- SlimeVolley-v0
  状態空間: Box(12)
  行動空間: MultiBinary(3)
- SlimeVolleyPixel-v0
  状態空間: Box(84, 168, 3)
  行動空間: MultiBinary(3)
- SlimeVolleyNoFrameskip-v0
  状態空間: Box(84, 168, 3)
  行動空間: Discrete(6)

状態空間の Box(12)は，各スライムの xy 座標，ボールの xy 座標，またそれらの速度といった情報を持つ．Box(84, 168, 3)では，画像データ（縦 84*横 168*RGB3 色）の情報を持つ．

行動空間の MultiBinary(3)はスライムの左右移動とジャンプの有無をそれぞれ 0 と 1 であらわしたものである．Discrete(6)は 0 がその場，1 が左移動，2 が左移動＋ジャンプといった具合にそれぞれ割り振ったものである．

いずれの環境も，扱うデータ形式が異なるのみで，ゲームの挙動やルール自体は全く同じである．

4.4 先行研究

本節では，本研究で Slime Volleyball Gym Environment を評価環境に用いた理由を先行研究と交えて述べる． Slime Volleyball Gym Environment は，ソースコードが Github 上で公開されおり，同時に様々な強化学習アルゴリズムを用いて学習を行った結果も掲載されている．現在はエージェント並列数を 96 とした PPO による学習によって得られたモデルが，最も高い平均スコア 0.435 を記録している．しかし，このゲームの理論上の最高スコアは 5 である．つまり一度もボールを落とさずに勝利した場合である．理論上の最高スコア 5 に対して現状最高で 0.435 であるため，スコアの伸びしろが大きいと考えられる．

前節で解説した 3 つの環境のうち，SlimeVolley-v0 は座標などを直接ネットワークに入力できる．しかし SlimeVolleyPixel-v0 と SlimeVolleyNoFrameskip-v0 は画像データを入力とするため学習が比較的難しく，実際に報告されているスコアも低い．また学習にかかる時間も長い．特に学習時間に関しては，前者は数時間で学習を終えているのに対し，後者は数週間かかったと報告されている[5].

このため，本研究では SlimeVolleyNoFrameskip-v0 の学習時間をスパコンによって短縮することを目的とし，さらに性能向上も図る．

5. スーパーコンピュータ

5.1 概要

スーパーコンピュータ（スパコン）について，明確な定義はないが，一般に同世代の計算機と比べて高い性能を持つ計算機のことを指す. そのほとんどが並列計算機である．近年の計算機資源の需要拡大に伴い，多くの大学や研究機関で運用されている．本研究においては名古屋大学情報基盤センターに設置されているスーパーコンピュータ「不老」を用いた．

スパコンの身近な利用例としては天気予報や地震のシミュレーション，また昨今では新型コロナウイルスに関連して飛沫の広がり方のシミュレーションなどが挙げられる．強化学習分野においても，Gorila[6]や A3C，Ape-X[7]で分散強化学習の有効性が示されて以降，多くの強化学習アルゴリズムで高い計算機性能を要求するようになったため，スパコンの需要が拡大している．

また近年，機械学習の高速化には GPU を用いることが多い．GPU は本来，グラフィックの処理を行うプロセッサであり，行列演算能力に長けているという特徴がある．機械学習の高速化には，この行列演算の高速化が重要であるため，GPU との相性がよい．GPU を主要な計算資源として用いるスパコンを GPU スパコンと呼ぶ.

5.2 Message Passing Interface

本研究の並列処理には MPI (Message Passing Interface) を用いた．MPI とは通信を行うプログラムのために標準化された規格の一つである．MPI によって，複数のプロセス間におけるデータの送受信を容易に行うことができ，並列動作が可能となる．また MPI を用いる利点として，移植の容易さとスケーラビリティの高さがあげられる．前者については，現在のスパコンは標準化された API に基づく高性能な MPI ライブラリを導入しているため, MPI を用いた同じプログラムは様々なスパコンで利用することができる．後者については，MPI は大規模なスパコンにおいて高い並列度で利用された実績が多い．そのため本研究でも移植性と性能を考慮し MPI を利用した．

なお本研究の開発言語は Python であるが, Python で MPI を呼ぶ際には mpi4py と呼ばれるライブラリが用いられることが多く，本研究でもこれを用いた．

6. 性能評価

6.1 概要

本章では Slime Volleyball Gym Environment を対象に, PPO におけるエージェント並列数の変化が与える学習時間とスコアへの影響について評価する．初めにエージェント並列数を変化させて学習時間の評価を行う．これは学習時間のスケール具合を見ることが目的である．そのため，先行研究で行っていた学習ステップより少ないステップで実験を行っている．その後，最も学習時間の短かったエージェント並列数によって，先行研究で行っていた学習ステップと同じステップ数学習し，学習時間と得られたモデルの性能評価を行った．

6.2 実験環境

名古屋大学情報基盤センター設置のスーパーコンピュータ「不老」Type II サブシステム及びクラウドシステムを使用した[8].「不老」Type II サブシステムの計算機構成を表 1 に示す．また「不老」クラウドシステムの計算機構成を表 2 に示す．（以降,「不老」Type II サブシステムのことを単に Type II,「不老」クラウドシステムのことを単にクラウドと呼ぶ）なお Type II は GPU スパコンである．クラウドには GPU は搭載されていないが, Type II と同様の CPU が 1 ノード当たり 2 倍（4 ソケット）搭載されている．ただし両システムは冷却性能などに違いがあるため，CPU 性能だけを比較した場合に完全に 2 倍の性能差となるとは限らない．

また先行研究との比較を行うため，使用するフレームワークやそのバージョンなどのソフトウェア環境については，可能な限り先行研究に揃えた．以下に本研究で使用したソフトウェア環境を示す.

- Python: バージョン 3.6.8
- gcc: バージョン 4.8.5

- OpenMPI: バージョン 4.0.3
- mpi4py: バージョン 3.1.3
- Tensorflow: バージョン 1.14
- stable-baselines: バージョン 2.10
- gym: バージョン 0.20.0
- slimevolleygym: バージョン 0.1.0

表 1 「不老」Type II サブシステムの構成

「不老」Type II サブシステムのハードウェア構成		
機種名		FUJITSU Server PRIMERGY CX2570 M5
計算ノード	CPU	Intel Xeon Gold 6230 × 2
	GPU	NVIDIA Tesla V100 × 4 （GPU 間接続：NVLINK2） （CPU-GPU 間接続：PCI-Express 3.0 (x16)）
	メインメモリ	DDR4 384GiB （メモリバンド幅：281.5GB/s）
	デバイスメモリ	HBM2 32GiB × 4 （メモリバンド幅：900 GB/s × 4）
ノード数		221 ノード
総理論演算性能		7.489PFLOPS (CPU 0.594PFLOPS, GPU 6.895PFLOPS)
総メモリ容量		メインメモリ 82.875TiB デバイスメモリ 28.288TiB
冷却方式		水冷

表 2 「不老」クラウドシステムの構成

「不老」クラウドシステムのハードウェア構成		
機種名		HPE ProLiant DL560
計算ノード	CPU	Intel Xeon Gold 6230 × 4
	メインメモリ	DDR4 384 GiB （メモリバンド幅：563.136 GB/s）
ノード数		100
総理論演算性能		537.6TFLOPS
総メモリ容量		37.5 TiB
冷却方式		空冷

6.3 実験結果（エージェント並列数と学習時間の比較）

本節では，エージェント並列数と学習時間の比較を行う．初めに Type II とクラウドでの学習時間の比較を行う．クラウド上で，1CPU コア当たり 1 エージェントを割り当てたところ，1 ノード当たり 64 エージェントまで動作させることができた．また Type II では 1 ノードにつき 4GPU 搭載されている．そのため，1GPU 当たり 1 エージェントを割り当てることで 1 ノード当たり 4 エージェントを動作させた．これを 16 ノード使うことでクラウドと同様に 64 エージェントを動作させた．学習時間の比較を図 2 に示す．ただしここではいずれの並列数においても経験を収集する総ステップ数を 500000 に揃えて実験を行っている．

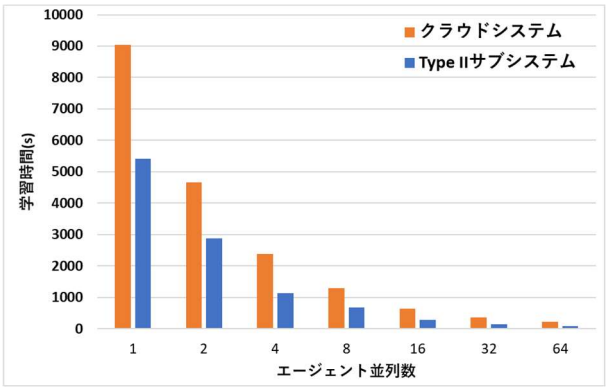

図 2 Type II とクラウドの学習時間の比較

図 2 より，Type II とクラウドのいずれも，エージェント並列数を増やすことで学習時間が短くなっていることが確認できる．また Type II ではクラウドと比較して 0.37～0.61 倍の学習時間と，約 2 倍の高速化となっている．ただしクラウドは 1 ノードで済んでいるのに対して Type II では最大 16 ノードを使用している．このことから，エージェント並列数をさらに増やす場合，ノード当たりの高速化の点においてはクラウドを用いる方が効率が良い．そこで，クラウドで一般に利用できる最大のノード数である 16 ノードを用いることでエージェント並列数を 1024 まで変化させて学習時間の比較を行った．結果を図 3 に示す．

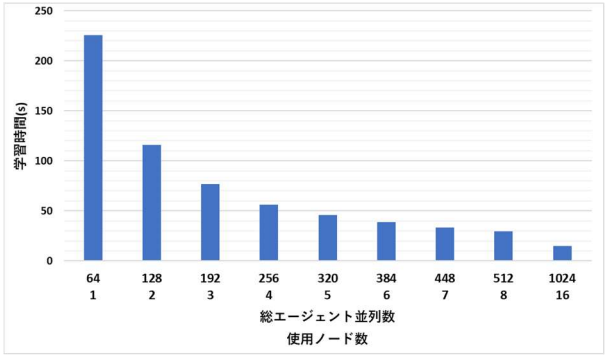

図 3 クラウドにおける並列数と学習時間の比較

図 3 から，64 からさらにエージェント並列数を増やしても学習時間が短くなることが確認できる．特に 1 ノード 64 並列から 16 ノード 1024 並列では約 15.5 倍の高速化がみられ，並列数の増加にほぼ比例して高速化している（台数効果が表れている）ことがわかる．

6.4 実験結果（従来スコアとの比較）

本節では，先行研究と本研究との学習時間とモデルの性能の比較を行う．前節で，本研究の実験環境では 1024 並列の時に学習時間が最も短くなることが示せたため，1024 並列で学習を最後まで行った．学習するステップ数について，先行研究では 2E+9 ステップ行っていたため，本研究でもこれに合わせた．

学習にかかった時間は 19 時間 20 分 29 秒であった．先行研究では学習に数週間を要したと報告されていたため，大幅な時間短縮となった．また学習によって得られたモデルを 1000 試合対戦させた際の平均スコアとスコアの分布を表 3 に示す．

表 3　先行研究と本研究のスコア比較

		先行研究（96並列）	本研究（1024並列）
スコア	5	0	0
	4	2	1
	3	23	34
	2	115	145
	1	269	313
	0	466	404
	-1	115	93
	-2	9	10
	-3	1	0
	-4	0	0
	-5	0	0
平均スコア		0.443	0.596
標準偏差		0.966	0.996

表 3 から，スコア-1 やスコア 0（制限時間により引き分け）となった試合は本研究の学習モデルの方が少なく，スコア 1 やスコア 2，スコア 3 などの勝利した試合は本研究の学習モデルの方が多いことがわかる．

6.5 考察

本研究において，エージェントの並列数増加が与えた影響について考察する．

まず学習時間の短縮について述べる．まず前提として，先行研究と本研究で学習する総ステップ数自体は変わっていない．しかし，エージェントの並列数が増えたことで 1 エージェント当たりが経験を収集するステップ数が分散された．これにより学習時間の短縮につながった．同時に，本研究で扱った PPO についてはネットワークの更新や勾配計算などの学習そのものにかかる時間よりも，経験の収集にかかる時間の方が大幅に大きかったため，GPU を用いても 2 倍程度しか高速化されなかったものと考えられる．ただしこれについては使用するフレームワークの問題も考えられる．

次に学習によって得られたモデルの性能向上について述べる．まず，エージェントの並列数が増えたことにより経験同士の自己相関が減り，学習が安定化したことが考えられる．3.3 章でも述べた通り，A3C ではエージェント並列数を増加させて性能が向上したことが報告されている．A3C では最大 16 並列までで実験を行い，並列数増加の有効性を示していたが，本研究の 1024 並列の場合においても同様にこの効果が表れたものと推察する．

またエージェント並列数が増えたことにより，乱数の面で学習モデルに影響を与えた可能性も考えられる．本研究で使用したフレームワークでは，環境ごとに別々の乱数がゲーム開始時に割り当てられる．これは実験に再現性を持たせるためであり，同じ乱数の種を使用すれば同じ環境を生成することができる．エージェント並列数が多い場合，それだけ多くの乱数が各環境生成時に割り当てられる．つまり乱数を引く回数が多くなるため，今回の実験では良い乱数の系列を引くことができたと考えられる．

良い乱数の系列を引くことが学習モデルの性能向上に影響する理由については次のように考察できる．まず，この問題における良い乱数の系列とは，良い経験データを得やすい乱数の系列のことを指す．良い経験データとは，これまで経験していなかったような状態や，大きな報酬を得た経験である．これらはネットワークの予測値との誤差が大きくなる原因となり，パラメータが大きく更新されることが期待できる．これにより学習が進みやすくなると考えられるため，良い乱数の系列を引くことはモデルの性能向上につながると考えられる．

7. まとめと今後の課題

本研究では Slime Volleyball Gym Environment を対象に，PPO によって学習を行い，性能評価を行った．スパコンを用いてエージェントの並列数を増やすことで，学習時間の短縮と，学習モデルの性能向上による平均スコアの増加がみられた．

今後の課題として以下の 3 点が挙げられる．一つ目にさらなる並列数の増加の検討である．本研究では先行研究の並列数 96 を 1024 へ増加させた．より多い並列数で学習した場合にスコアに与える影響を評価する．二つ目に PPO 以外のアルゴリズムの検討が挙げられる．エージェント並列数の増加によるスコアの向上には限度があると考えられるため，さらなるスコアの向上にはアルゴリズム自体の変更が必要である．PPO よりも後に考案された強化学習アルゴリズムで，まだ Slime Volleyball Gym Environment において性能評価がなされていない R2D2[9]や Agent57[10]などによってさらなるスコアの向上を図る．三つ目にほかの評価環境においても同様の性能評価を行うことが挙げられる．特に，強化学習アルゴリズムの性能評価に一般に用いられている Atari ゲーム環境などにおいても，今回のような 1000 を超えるエージェント並列数が有効であるのかを確かめる余地がある．

参考文献

[1] Schulman, J., Wolski, F., Dhariwal, P., et al. , Proximal Policy Optimization Algorithms, arXiv preprint arXiv:1707.06347, 2017.
[2] Mnih, V., Badia, A., Mirza, M., et al. , Asynchronous Methods for Deep Reinforcement Learning, Proc. ICML, pp. 1928-1937, 2016.
[3] David Ha：Slime Volleyball Gym Environment, GitHub（オンラ

イン），入手先<https://github.com/hardmaru/slimevolleygym>（参照 2022-02-19）
[4] OpenAI：Gym, OpenAI（オンライン），入手先< https://gym.openai.com/>（参照 2022-02-19）
[5]David Ha：How many timestep experiments have you done with ppo_pixel? #16 , GitHub（オンライン），入手先< https://github.com/hardmaru/slimevolleygym/issues/16>（参照 2022-02-19）
[6] Nair, A., Srinivasan, P., Blackwell, A., et al., Massively Parallel Methods for Deep Reinforcement Learning, Presented at ICML, 2015.
[7] Horgan, D., Quan, J., Budden, D., et al., Distributed Prioritized Experience Replay, Proc. ICLR, 2018.
[8] 名古屋大学情報連携推進本部：スーパーコンピュータ「不老」紹介，名古屋大学情報連携推進本部（オンライン），入手先< https://icts.nagoya-u.ac.jp/ja/sc/overview.html >（参照 2022-02-19）
[9] Kapturowski, S., Ostrovski, G., Quan, J., RECURRENT EXPERIENCE REPLAY IN DISTRIBUTED REINFORCEMENT LEARNING, Proc. ICLR, 2019.
[10] Puigdomènech, A., Piot, B., Kapturowski, S., Agent57: Outperforming the Atari Human Benchmark, arXiv preprint arXiv: 2003.13350