2048への方策勾配法の適用

山下修平[1,a)]　金子 知適[2,b)]

概要：本稿では 2048 という確率的ゲームを題材に強化学習における方策勾配法の性能について研究する．強化学習はエージェントが与えられた環境において試行錯誤を通じて最適な方策を学習するための手法である．強化学習には大きく分けて状態や行動の価値関数を学習することで最適な方策を見つける手法と, 方策勾配定理に従って直接方策を改善していく手法の 2 つがある．2048 においては Szubert らが TD-afterstate 学習を提案して以来, 主に前者のアプローチを主流としてハイスコアが更新されてきた．本研究では方策勾配法による方策の学習が 2048 においても可能であることを示す．さらにエージェントに与える報酬は専らゲームスコアが使われてきたが, より長くエピソードが続くことを期待して 1 ステップごとに+1 としても同等以上の成果が得られることを示す．

キーワード：強化学習, 2048, 方策勾配法, Proximal Policy Optimization

Application of the policy gradient method to 2048

Shuhei Yamashita[1,a)]　Tomoyuki Kaneko[2,b)]

Abstract: This paper studies the effectiveness of policy gradient methods on a stochastic game 2048. Reinforcement learning is a method in which an agent learns an optimal policy through trial and error in a given environment. There are mainly two ways in reinforcement learning to find an optimal policy: one is by learning state or action value functions, and the other is by directly improving its policy according to the policy gradient theorem. In 2048, the high scores achieved by AI agents have been updated mostly with the former approach since Szubert presented TD-afterstate learning. In this paper, we show that an agent can learn its policy by policy gradient method too. Also, games scores have been used exclusively as the reward to train agents until now. However, we show that the same or better results can be obtained if the agent is given +1 reward for each step so that an agent prefers longer episodes more.

Keywords: Reinforcement Learning, 2048, Policy Gradient Method, Proximal Policy Optimization

1. はじめに

強化学習は与えられた環境において, エージェントに試行錯誤を通して報酬を最大化するような一連の行動 (最適方策) を学習させるための枠組みである．これに深層学習の技術を取り入れた深層強化学習は Deep Q Network (DQN) [1] の登場以来大きな進歩を遂げており, ゲーム AI の分野においても人間の能力を大きく上回るエージェントの開発に成功している．DQN は状態行動価値関数を学ぶ Q 学習を基本とする手法である．また Proximal Policy Optimization (PPO) [2] はパラメタライズされた方策を直接学習する方策勾配法の手法で, ロボット制御や atari ゲームなど幅広く使われている現在主流の深層強化学習アルゴリズムの

[1] 東京大学教養学部学際科学科
Department of Interdisciplinary Sciences, The University of Tokyo
[2] 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
a) yamashita-shuhei@g.ecc.u-tokyo.ac.jp
b) kaneko@graco.c.u-tokyo.ac.jp

1 つである．

本研究の対象のゲームは 2048 である．2048 は，2014 年に Gabrirle Cirulli によって公開されたパズルゲームである [3]．ルールは非常に簡単で 1 ステップでの行動数は 4 と少ないが，高得点を獲得するためには長期的に正しい行動を取り続ける必要がある．さらにゲームが進むにつれて難易度が上がるという特徴を持っている．

これまで 2048 においては DQN のように価値関数を学習することで方策の改善を行う手法は成果を上げてきたが，PPO を含めて方策勾配法によって成果を上げた例は存在しない．本稿では工夫を施すことで PPO によって 2048 においても問題なく学習ができることを実験により示す．さらにエージェントに与える報酬は専らゲームスコアが使われてきたが，より長くエピソードが続くことを期待して 1 ステップごとに+1 としても同等以上の成果が得られることを実験により示す．

2. 2048

2048 は 4×4 の 16 マスの盤面上で遊ばれるゲームである．16 マスの内 2 マスに 2 か 4 のタイルがそれぞれ 90%，10%で置かれた状態で始まる．プレイヤは上下左右いずれかの方向を選択する．すると各数字タイルはその方向に移動し，盤の外枠もしくは他のタイルにぶつかったところで止まる．移動した結果 2 つの同じ数字を持つタイルが衝突するとこれらは合体してその合計の数へ変化し，プレイヤはその数値を報酬として得る．なお一度合体したタイルは，プレイヤの次の行動まで合体しない．その後に空白のマスからランダムに選択されたマスに 90%の確率で 2 が，10%の確率で 4 が置かれる．図 1 に右を選択した場合の遷移の例を示している．なお行動後新しい数字が出現する前の状態を afterstate と呼ぶ．プレイヤは 4 通りの方向の中で 1 つ以上のタイルが移動する，もしくは同じ数字同士が衝突することで盤面が変化するような方向しか選択することができない．いずれの方向も選択できなくなるとゲームは終了する．プレイヤの目標はなるべく多くの報酬を獲得することである．ゲームの一般的なゴールは 2048 のタイルを作ることだが，当然それ以降も続けることができる．また

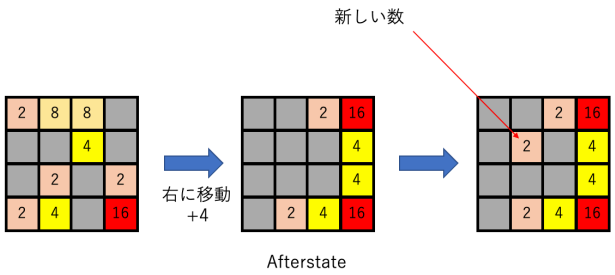

図 1 右を選択した場合の変化の例

盤面に大きな数字のタイルが初めて出現する際のゲームスコアとそれまでに必要な行動の数の目安は表 1 のように知られている [4]．

表 1 大きなタイルが初めて出現する際のスコアと必要な行動数の目安 [4]

タイル	スコア	行動数
1024	9000	480
2048	20000	950
4096	44000	1900
8192	97000	3800
16384	210000	7500
32768	450000	15000

3. 強化学習

本章では強化学習の概要について簡単に解説する．詳細は Sutton and Barto (2018) [5] を参照されたい．強化学習は与えられた環境において，エージェントに試行錯誤を通して報酬を最大化するような一連の行動 (最適方策) を学習させるための枠組みである．強化学習の問題は (S, A, p, r) からなるマルコフ決定過程で定式化される．ここで S は状態集合，A は行動集合，$p : S \times S \times A \to [0, 1]$ は状態遷移確率，$r : S \times A \to R$ は報酬関数を表す．さらにある状態 s からある行動 a をとる確率 $\pi(s|a)$ を方策と呼び，エージェントはそれぞれの時刻 t において方策 π に従って行動を選択することを繰り返す．強化学習の目的はゲームの累積報酬の期待値 $\mathbb{E}_\pi[\sum_{t=0}^{T} r_t]$ を最大化するような方策 (最適方策) を見つけることである．最適方策を見つけるための手法としては大きく分けて状態の価値関数を学習する手法 (価値ベースな手法) と，方策勾配定理に従って直接パラメタライズした方策を改善していく手法 (方策勾配法) の 2 つがある．

本研究は 2048 を方策勾配法によって攻略することを目指しているので方策勾配法について簡単に紹介する．方策勾配法では，方策を π_θ とパラメータ θ をつかって表現し，このパラメータを変化させることでより良い方策を実現することを目指す．単純な方策勾配法では以下の目的関数を最大化するように方策を更新する．

$$L(\theta) = \mathbb{E}_t[\log \pi_\theta(a_t|s_t) A_t]$$

s_t, a_t はそれぞれタイムステップ t 時点での状態，行動を表し，A_t は advantage の推定値を表す．advantage の推定値 A_t は t 時点でとった行動の良さの推定量である．そのため直感的には advantage の推定値が高い行動，すなわちより良い行動をとる確率を上げるように θ を更新すれば目的関数が改善されると解釈できる．この advantage の推定値として古典的な REINFORCE アルゴリズムは単純にエピソードの t 以降の累積報酬 (Monte Carlo return) G_t を用いる．しかしこれは環境によっては分散が大

きく学習が安定しないので, 価値関数 V も同時に学習し $A_t = G_t - V(s_t)$ などとすることで分散を減らす努力が為される. それと同時に G_t の推定にも価値関数を用いて n step return ($G_{t:t+n} = r_t + \gamma r_{t+1} + \cdots \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n})$) を使うことで分散を抑えることも行われる. このような方策と状態価値の両方を関数として学習する手法では方策関数は Actor, 価値関数は Critic と呼ばれ, 一般に Actor Critic アルゴリズムと言う.

4. Proximal Policy Optimization

Proximal Policy Optimization (PPO) [2] は方策勾配法の一種で方策の更新に制限を加えたアルゴリズムであり, 強化学習において現在主流の方法の 1 つである. 単純に方策勾配定理に従った学習を同じデータで何度も行うと過剰にパラメータを更新してしまい, 方策の破綻につながることがある. そこで PPO は目的関数を以下のように変更することでこの問題に対処する.

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

ここで $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\mathrm{old}}}(a_t|s_t)$ であり, あるミニバッチの学習前の方策の出力 $\pi_{\theta_{\mathrm{old}}}(a_t|s_t)$ と現在の方策の出力 $\pi_\theta(a_t|s_t)$ の比を表している. PPO はこの比が大きく変化しないように目的関数を工夫することで学習時のパラメータの急激な更新を防ぎ安定した方策の改善と効率的な学習の両立を目指している.

5. 2048 に関する先行研究と考察

これまで 2048 を攻略するエージェントとしては expectimax 探索やモンテカルロ木探索を使ったものなど様々な研究されてきた. 中でも最も成功したアプローチは Szubert らが提案した TD-afterstate 学習 [6] である. これは盤面評価関数により盤面の価値 V を計算し, 行動後の即時報酬と afterstate の価値の合計が最も大きな行動を選ぶ. そして afterstate の価値同士の TD 誤差を最小化することでパラメータの更新を行うアルゴリズムである. すなわち現在の状態が s_t で行動 a_t を選択した結果, afterstate s'_t を経由して報酬 r_t を獲得し s_{t+1} へ遷移したとする. さらに s_{t+1} から a_{t+1} を選択し afterstate s'_{t+1} を経由して報酬 r_{t+1} を獲得し s_{t+2} へ遷移したとする. このとき $V(s'_t) \leftarrow V(s'_t) + \alpha(r_{t+1} + V(s'_{t+1}) - V(s'_t))$ という更新式でパラメータを更新する. ただし α は学習率である. この盤面評価関数として Szubert らは n-tuple network [6] を, 松崎 [4] はニューラルネットワークを関数近似器として用いていずれも高い成果を出している.

Szubert らは n-tuple network を使って Q 学習についても実験をしたが TD-afterstate 学習に比べるとかなり悪い成果しか出せなかったことを述べている [6]. また Guei らは DQN [1] の性能を調査したが 100,000 ゲーム学習後もゲームクリアの 20,000 点にも安定して届いていない [7].

ここで Q 学習と TD-afterstate 学習の違いについて考察する. 評価関数としてニューラルネットワークを使った場合を考えるが n-tuple network を使った場合についても同様である. DQN ではニューラルネットワークは状態 s_t と行動 a_t という入力に対して, 次の状態 s_{t+1} の価値 $V(s_{t+1})$ を正しく推定できるように学習する必要がある. 一方 TD-afterstate 学習においては状態 s_t から行動 a_t をとった結果遷移する afterstate s'_t を先に計算した上で, s'_t を入力とするのでニューラルネットワークはその価値を正しく推定できるように学習するだけでよい. すなわちニューラルネットワークに対して状態遷移に関する情報を事前に抽出して与えることで学習が容易になっているのだと考えられる. 一般に状態遷移関数が決定的でない環境においては行動選択後の状態が確率的であるため最適状態価値関数が分かっていたとしても最適方策を導くことができない. そのため状態行動価値関数を学習する Q 学習が有力となる. 2048 も行動後に出現する新しい数字およびその位置は確率的であるため状態遷移関数は決定的でない. しかし afterstate s'_t に至るまでは決定的であるためどのような afterstate に遷移するかは事前に計算することが可能となる. よって TD-afterstate 学習は環境の決定的な部分を最大限に利用することで成功したアルゴリズムであると言える.

価値ベースの手法である TD-afterstate 学習が成功したのに対して, 方策勾配法によって 2048 の攻略に成功した例はこれまでになく, 難しいと考えられていた. しかしこれは Q 学習と TD-afterstate 学習の比較から, 方策関数を状態 s_t と行動 a_t に対して行動確率 $\pi(a_t|s_t)$ を正しく推定できるように学習させるところに問題があると考えられる. すなわち afterstate を考慮した方策を設計すればよく, 方策勾配法のアルゴリズムが 2048 に適していないということではないと考える. このことを踏まえて次章では Afterstate PPO を提案する.

6. 提案手法

本節では 2048 に PPO を適用するための手法としての Afterstate PPO について説明する. Afterstate PPO は Actor Critic のフレームワークであり, 状態を入力として行動確率を出力する Actor と状態の価値を計算する Critic を持っている. Actor と Critic はいずれも 1 つの afterstate を受け取ってそのスカラーの評価値 (preference) を計算するニューラルネットワーク f_{Actor} と期待報酬を推定する f_{Critic} を持っている. Actor はある状態 s から上下左右 4 つの行動をとった際に得られる afterstate $s_{\mathrm{up}}, s_{\mathrm{down}}, s_{\mathrm{right}}, s_{\mathrm{left}}$ を計算し, これらの preference を f_{Actor} で計算する. $[f_{\mathrm{Actor}}(s_{\mathrm{up}}), f_{\mathrm{Actor}}(s_{\mathrm{down}}), f_{\mathrm{Actor}}(s_{\mathrm{right}}), f_{\mathrm{Actor}}(s_{\mathrm{left}})]$

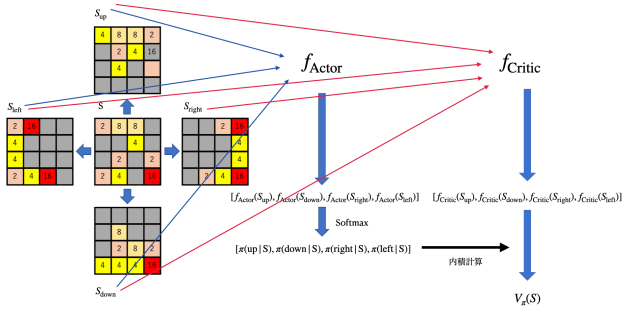

図 2 Afterstate PPO における Actor と Critic

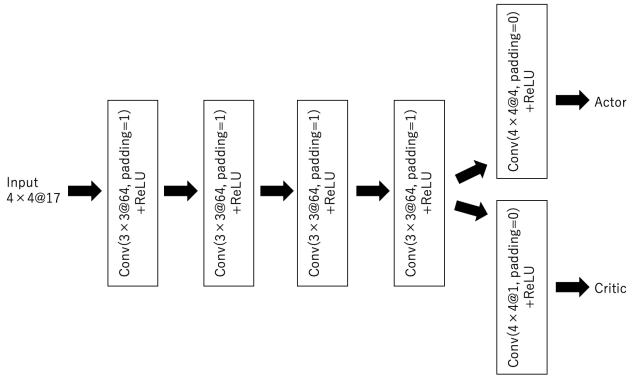

図 3 Normal PPO のニューラルネットワーク

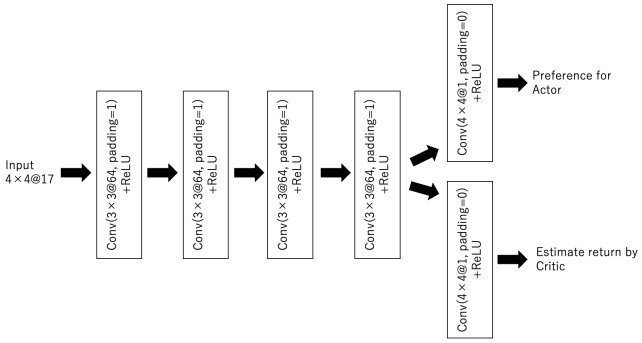

図 4 Afterstate PPO のニューラルネットワーク

の softmax をとることでそれぞれの行動をとる確率 $\pi(\text{up}|s), \pi(\text{down}|s), \pi(\text{right}|s), \pi(\text{left}|s)$ を得る． Critic も Actor と同じように f_{Critic} によってそれぞれの方向の afterstate の期待報酬を計算し，それぞれに対する Actor が出力する行動確率で重み付け平均を計算することで $V_\pi(s)$ を計算する．図 2 に Actor と Critic の動きについて示している．こうすることで afterstate を最大限に利用した Actor Critic のモデルを構築することができる．

Actor の学習で必要な advantage の推定値には，PPO を使う場合に標準的な Generalized Advantage Estimate (GAE) [8] を用いる．GAE は，n-step return を様々な n について，加重平均をとったものである．Critic は 1 ステップ TD 誤差を最小化するように更新する．すなわち Critic は $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ を最小化し，Actor は $A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}$ として式 (1) の目的関数を最大化するようにパラメータを更新する．

またこれまで即時報酬はゲームのプレイで発生するスコアの増分とする (以降 score reward と呼ぶ) ことが常識だった．実際 2048 は高得点を獲得することがゲームの目標であるためスコアの増分を即時報酬にするのは自然な考え方である．しかし 2048 はゲームオーバーにならずゲームが続くほど高得点を獲得することが期待できるゲームであるため，即時報酬を 1 ステップごとに+1 とすることが考えられる．以降これを step reward と呼ぶことにし，score reward との比較を実験にて示す．

7. 実験

本章では Afterstate PPO の能力，およびエージェントに与える score reward か step reward かでどのように異なるかを実験により評価する．7.2 節では Afterstate PPO と Afterstate を使わない PPO (Normal PPO) を比較する．7.3 節では reward を score reward とした場合と step reward とした場合での比較を先行研究の TD-afterstate 学習と Afterstate PPO で検証する．これらを踏まえて 7.4 節では reward を 1 ステップごとに+1 とした時の Afterstate PPO を 10 万ゲーム学習させて最終評価する．実験に使用したプログラムは `https://github.com/shuymst/2048_GPW2021` で利用可能である．

7.1 ニューラルネットワークの詳細

本研究で使用するニューラルネットワークの構造について述べる．なお全体を通して 2 次元の盤面は Guei ら [7] が提案したエンコーディングを施すことで，4×4, 17 チャネルの 3 次元に変換してニューラルネットワークに入力する．すなわち n 番目のチャネルは元の盤面で 2^n が存在する位置が 1，それ以外は 0 となるようにする．ただし 0 番目のチャネルは $2^0 = 1$ ではなく元の盤面で空白の位置に 1 が立つ．

単純な PPO (Normal PPO と呼ぶ) は素直に状態から 4 方向への確率を出力するニューラルネットワークを Actor，状態から状態価値を出力する Critic を持つ．Normal PPO と Afterstate PPO の Actor Critic のニューラルネットワークの構造はそれぞれ図 3, 図 4 に示す通りである．ただし図中の Conv は畳み込み層を表し，例えば 3×3@64, padding=1 とはカーネルサイズが 3×3 で 64 チャネル，パディングが 1 という意味である． また学習時にはそれぞれの畳み込み層と ReLU の間に Batch Normalization を挟んでいる．

7.3 節では score reward と step reward の比較を先行研究の TD-afterstate 学習においても行う．この際の価値関数を近似するニューラルネットワークは松崎が文献 [4] で使用した CNN22 を採用した．

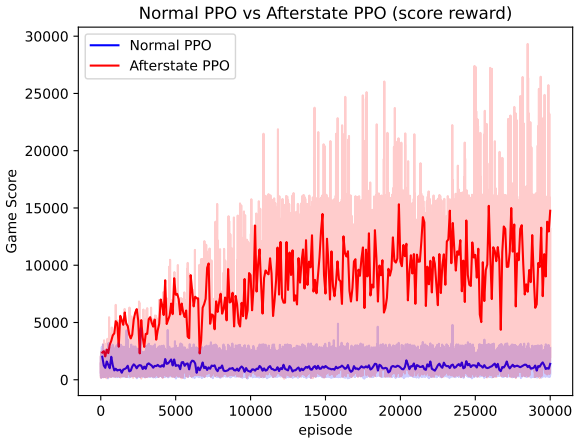

図 5 score reward での Normal PPO と Afterstate PPO の比較

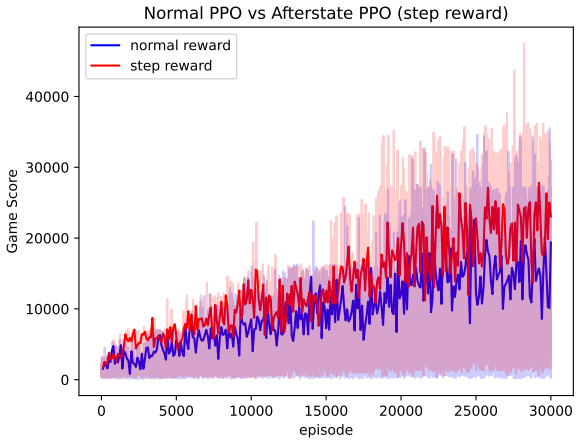

図 7 Afterstate PPO における score reward と step reward の比較

Afterstate PPO および TD-AFTERSTATE 学習の実験に使用した最適化アルゴリズムやハイパーパラメータに関する詳細は付録 A.1 を参照されたい.

7.2 Normal PPO と Afterstate PPO の比較

本節では Normal PPO と Afterstate PPO の性能を比較する.

score reward と step reward の両方で Normal PPO と Afterstate PPO の学習を 30,000 ゲーム行なった結果を図 5, 図 6 に示す. 実験時間の制約から長いゲーム数での学習を観察することができなかったため, 学習の経過と先行きがある程度判断できる 30,000 ゲームで打ち切ることにした. グラフは毎ゲームのスコア (薄い色のグラフ) と 100 ゲームごとに評価モードで 10 回プレイした平均スコア (濃い色のグラフ) をプロットした. 通常エージェントは Actor が出力する確率分布に従って行動を選択するが, 評価モードでは Actor が最も高い確率を出力するような行動を選択する. これは DQN が評価時には ϵ-greedy の ϵ を 0 にした greedy な状態で評価を行うので, それに倣う形とした. 以降の実験結果のグラフについても同様である.

図 5, 図 6 の赤い線を見ると Afterstate PPO は score reward の場合も step reward の場合も学習を進めており, 20,000 点越えも達成している. 一方で score reward の Noraml PPO は全く学習が進んでおらず, step reward の Normal PPO はそれに比べると少し学習をしたのが見て取れるが, ゲームクリアの目安となる 20,000 点とは程遠く停滞してしまっている.

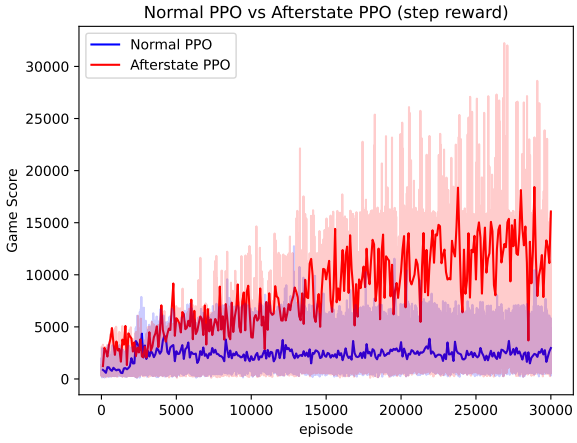

図 6 step reward での Normal PPO と Afterstate PPO の比較

7.3 score reward と step reward の比較

本節では score reward と step reward の比較を行う. まず Afterstate PPO での両者の比較を図 7 に示す.

次に TD-AFTERSTATE learning での step reward の実験結果を図 8 に示す. 実験時間の制約から score reward との比較を示すことができなかったが, 順調に学習をしている様子が見て取れる. なお図 7 と 7.4 節の図 9 の濃い青の実戦は, 訓練中のエピソードの移動平均 (30 回分) である.

複数回での平均をとるような実験を行うことができなかったため現時点では優劣について断言することはできないが, step reward でも十分に学習を進められると思われる.

7.4 Afterstate PPO (step reward) の評価

最後に step reward での Afterstate PPO を 100,000 ゲーム学習させて現時点での最終評価を行った. 毎ゲームのスコアとその 30 ゲームでの移動平均をとったグラフを図 9 に示す. ブレは大きいが学習するほど獲得スコアが上がっていることがわかる. 実験時間の制約から 100,000 ゲーム

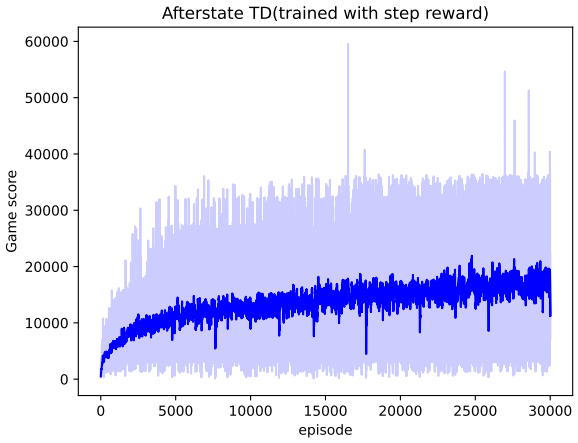

図 8 TD-AFTERSTATE learning における step reward での学習

表 2 盤面の最大値とその回数

最大値	回数
256	1
512	5
1024	31
2048	56
4096	7

で打ち切ったが, 学習を続ければまだ性能が向上しそうな様子が伺える.

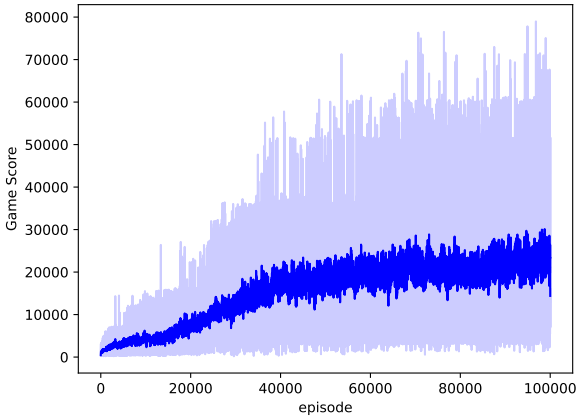

図 9 step reward での Afterstate PPO(100,000 ゲーム)

学習後のモデルで 100 回ゲームをプレイした結果平均は 26,861.84 点となった. ゲーム盤面の最大値の内訳を表に示す.

8. まとめと今後の課題

本稿では方策勾配法で 2048 を攻略するための手法として Afterstate PPO を提案し十分な学習性能を持つことを示した. さらに学習時の報酬を 1 ステップごとに+1 とする step reward でも通常のゲームスコアを報酬とする score reward と同等以上の学習ができることを示した. これまでの先行研究においても報酬を step reward に変更するだけで学習が改善される可能性は十分にあると考えられる. また付録 A.1.3 で示しているように, 本稿では実験時間の制約から比較的小さなニューラルネットワークでの実験を行った. 一般にニューラルネットワークを大きくするほど表現力の向上によって性能は上がるので, より大きなニューラルネットワークを使い訓練時間をより増やすことで Afterstate PPO によって TD-AFTERSTATE 学習による先行研究の成果を越える可能性は十分にあると考えられる.

今後はより大きなニューラルネットワークでの実験や score reward と step reward のより詳細の比較, 訓練期間を長くした実験を行いたいと考えている. それによって TD-AFTERSTATE 学習と Afterstate PPO の綿密な比較を議論できると考えている. また 2048 においては同程度の累積報酬を獲得するための一連の行動として複数の選択肢が存在することが多い. 同じ累積報酬を獲得できる 2 つの状態では多くの選択肢を持つ状態の方が良い状態だと考えられる. よってエントロピーを考慮した学習について研究することも興味深いテーマだと考えている.

参考文献

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D.: Human-level control through deep reinforcement learning, *Nat.*, Vol. 518, No. 7540, pp. 529–533 (online), DOI: 10.1038/nature14236 (2015).

[2] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O.: Proximal Policy Optimization Algorithms (2017). https://arxiv.org/abs/1707.06347.

[3] Cirulli, G.: 2048, available from ⟨http://gabrielecirulli.github.io/ 2048/⟩ (2014).

[4] Matsuzaki, K.: Developing Value Networks for Game 2048 with Reinforcement Learning, *Journal of Information Processing*, Vol. 29, pp. 336–346 (online), DOI: 10.2197/ipsjjip.29.336 (2021).

[5] Sutton, R. S. and Barto, A. G.: *Reinforcement Learning: An Introduction*, The MIT Press, second edition (2018).

[6] Szubert, M. and Jaśkowski, W.: Temporal difference learning of N-tuple networks for the game 2048, *2014 IEEE CIG*, pp. 1–8 (online), DOI: 10.1109/CIG.2014.6932907 (2014).

[7] Guei, H., Wei, T., Huang, C.-B. and Wu, I.-C.: An Early Attempt at Applying Deep Reinforcement Learning to the Game 2048, *Workshop on Neural Networks in Games* (2016).

[8] Schulman, J., Moritz, P., Levine, S., Jordan, M. and Abbeel, P.: High-Dimensional Continuous Control Using Generalized Advantage Estimation (2018).

表 A·1 Afterstate PPO のハイパーパラメータ

ハイパーパラメータ	値
最適化手法	SGD
学習率	0.002
momentum	0.9
weight decay	0.0001
割引率 (γ)	0.99
割引率 (λ)	0.99
エポック数	30

表 A·2 TD-AFTERSTATE 学習のハイパーパラメータ

ハイパーパラメータ	値
最適化手法	Adam
学習率	0.0003
割引率 (γ)	0.99
ϵ	0.1

表 A·3 各ニューラルネットワークの重みの数

モデル	重みの数
Normal PPO	125,509
Afterstate PPO	122,434
CNN22(文献 [4] のモデルの著者による再現)	2,902,273

付　　録

A.1 実験の詳細

A.1.1 実験に使用した環境

本稿で行った実験には, 次の性能の計算機およびソフトウェアを使用した.

- OS : Ubuntu 20.04.3 LTS (Focal Fossa)
- CPU : AMD Ryzen Threadripper 3970X
- GPU : GeForce GTX2080Ti 1 枚
- Python : バージョン 3.8.10
- Pytorch : バージョン 1.7.1

A.1.2 ハイパーパラメータ

Afterstate PPO は学習にあたって, ゲーム終了時に 4,000 以上のサンプルが集まっていればパラメータの更新を行う. ただし前回の更新から 10 ゲーム以上経過していない場合は更新を行わない. ハイパーパラメータとその値を表 A·1 に示す.

次に TD-AFTERSTATE 学習のハイパーパラメータとその値を表 A·2 に示す. 学習時には探索を促進するため確率 ϵ でランダムに行動を選択する ϵ-greedy を用いた. また松崎は計算効率の向上のためにゲームのプレイとパラメータの更新を並列に実装する工夫をとっていたが, 今回は score reward と step reward の検証が目的だったため簡略化しゲームの終わりごとに 1 ゲームのサンプルを 1 つのバッチとしてパラメータの更新を行う実装とした.

A.1.3 ニューラルネットワークの重みの数

本稿の全ての実験で使用したニューラルネットワークのモデルの重みの数をまとめたものを表 A·3 に示す.