深層強化学習における擬似的な行動による中間フレームの有効活用

橋本 大世[1,a)]　鶴岡 慶雅[1]

概要：深層強化学習の多くの設定ではエージェントが行動を取る際, 一度選んだ行動を何度か繰り返し, 次の行動決定時まで状態は観測しないことが一般的である. これは action repeat または frame skip と呼ばれる. 行動を繰り返すこの技法にはいくつかの利点があるが, 行動を繰り返す間のデータ (中間フレーム) は実質的に捨てられてしまう. 学習データ量は action repeat の長さに反比例するため, これは学習のサンプル効率に悪影響となりうる. 本研究では, 擬似的な行動という概念を導入することでこの問題を軽減する, シンプルでありながら有効な手法を提案する. 提案手法の要点は, 擬似的な行動を考えることで, action repeat 間の遷移データを学習に利用できるようにすることである. 連続制御タスクにおける擬似的な行動は, 行動を決定する時刻をまたぐ行動系列の平均として得ることができる. 一方, 離散制御タスクにおける擬似的な行動は, 行動の埋め込み表現から計算することができる. この手法は, Q 関数の学習を伴う任意のモデルフリー強化学習手法と組み合わせることができ, 汎用的である. 実験では, OpenAI Gym の連続制御タスク, 離散制御タスクの両方で提案手法の有効性を検証した.

Utilizing Skipped Frames in Deep Reinforcement Learning via Pseudo-Actions

Taisei Hashimoto[1,a)]　Yoshimasa Tsuruoka[1]

Abstract: In many deep reinforcement learning settings, when an agent takes an action, it repeats the same action a predefined number of times without observing the states until the next action-decision point. This technique of action repetition has several merits in training the agent, but the data between action-decision points (i.e., intermediate frames) are, in effect, discarded. Since the amount of training data is inversely proportional to the interval of action repeats, they can have a negative impact on the sample efficiency of training. In this paper, we propose a simple but effective approach to alleviate to this problem by introducing the concept of pseudo-actions. The key idea of our method is making the transition between action-decision points usable as training data by considering pseudo-actions. Pseudo-actions for continuous control tasks are obtained as the average of the action sequence straddling an action-decision point. For discrete control tasks, pseudo-actions are computed from learned action embeddings. This method can be combined with any model-free reinforcement learning algorithm that involves the learning of Q-functions. We demonstrate the effectiveness of our approach on both continuous and discrete control tasks in OpenAI Gym.

1. はじめに

強化学習を実社会において利用できるようにするためには, 少ないデータから安定して学習する手法が必要不可欠である. アルゴリズムの改良によりこの問題に取り組んだ研究は数多くあるが, それらの多くは収集したデータをいかにうまく活用するかという点に主眼を置いている [1], [2], [3]. 一方 Kostrikov らは, 一般的にコンピュータビジョンなどの分野で用いられているデータ拡張により, 画像から効率よく方策を学習できることを示した [4]. この

[1] 東京大学大学院情報理工学系研究科電子情報学専攻
Department of Information and Communication Engineering, Graduate School of Information Science and Technology, The University of Tokyo
[a)] hashi@logos.t.u-tokyo.ac.jp

研究により, 深層強化学習は他の分野と同様に過学習による悪影響を受けやすく, またデータを増やすことで過学習を軽減できることがわかった. この発見に動機づけられ, 我々はエージェントの学習データ量を増やす方法を探し, action repeat という深層強化学習で広く用いられている技法に着目した.

強化学習の一般的な設定として, エージェントが一旦行動を選択するとその行動は固定された回数繰り返される. 行動を繰り返すこの技法は, 学習される方策の質に大きく影響することが知られている [5], [6]. 強化学習における離散制御タスクの代表的なベンチマークである Atari 2600 [7] では, 繰り返しの回数を 4 に設定することが多い [7], [8]. 連続制御タスクの代表的なベンチマークである DeepMind control suite [9] では環境ごとに異なる値が使われることが多いが, 2 から 4 程度が一般的である [10], [11].

action repeat は学習の安定化 [12] や探索の促進 [13] などのために広く用いられている. しかし, action repeat の問題点として, 行動を繰り返す間のデータは十分に活用されていないことが挙げられる. DeepMind control suite では通常 action repeat 間のデータは全く使用されていない. Atari 2600 では action repeat 間の最後の 2 フレームの最大値プーリングが使用されるが, これは偶数フレーム, 奇数フレームしか現れないオブジェクトを見落とさないようにするためである. 環境のステップ数が一定である限り, 学習データ量は action repeat の長さに反比例するため, action repeat が長い場合に非常に多くのデータが捨てられてしまう. そこで本研究では, action repeat 間の状態遷移から得られる情報を有効活用することを目的とする.

提案手法は, 行動の平均値で表される擬似的な行動が action repeat の回数だけ繰り返されたと想定することで, 行動を繰り返す間のデータを利用可能にする. この手法は連続行動空間には直接適用できるが, 離散行動空間では意味のある行動の平均を得るために工夫が必要である. そこで我々は, 行動の埋め込み表現を学習してその平均を用いるという方法を考案した.

この論文の貢献は次の 3 点である. (1) action repeat 間の遷移データを利用することで, 連続制御タスクでデータ量を増やす手法を提案する. (2) 行動の埋め込み表現を学習することで, 提案手法を離散制御タスクに拡張する. (3) 提案手法と Deep Q-learning, Soft Actor-Critic を組み合わせることで, いくつかの連続制御タスク, 離散制御タスクでパフォーマンスを向上させた.

2. 背景

2.1 強化学習

強化学習では, マルコフ決定過程 (Markov Decision Process, MDP) [14] で表される環境の中でエージェントが方策を学習する. MDP は $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ で定義される. $\mathcal{S}$ は状態空間, $\mathcal{A}$ は行動空間である. p: $\mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ は遷移確率であり, $p(s_{t+1}|s_t, a_t)$ は状態 $s_t \in \mathcal{S}$ において行動 $a_t \in \mathcal{A}$ を取った際に次状態が s_{t+1} になる確率を表す. r: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ は報酬関数であり, 状態 s_t において行動 $a_t \in \mathcal{A}$ を取った際に得られる報酬を表す. p, r はエージェントには与えられず, 必要であればデータから推定する必要がある. $\gamma \in [0, 1)$ は割引率である. 強化学習の目的は, 割引された累積報酬 $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \,|\, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)\right]$ を最大化する方策 $\pi(a_t|s_t)$ を学習することである.

本研究では, 画像を観測として実験を行った. この場合, 環境は部分観測マルコフ決定過程 (Partially Observable Markov Decision Process, POMDP) として定式化される. なぜなら, 物体の速度など, 1 枚の画像からは得られない情報があるからである. POMDP は $(\mathcal{O}, \mathcal{A}, p, r, \gamma)$ で表され, $\mathcal{O}$ は観測空間である. 通例に従い [15], 直近の画像入力をいくつか積み重ねて状態 $s_t = \{o_t, o_{t-1}, o_{t-2}, \ldots\}$ とすることで, POMDP を MDP として扱う.

2.2 Deep Q-learning

Deep Q-learning [15] は離散制御タスクにおける深層強化学習の代表的なアルゴリズムである. 状態 s において行動 a を取った後に得られる累積報酬の期待値は Q 値と呼ばれ, 写像 $Q(s, a)$ は Q 関数と呼ばれる. Deep Q-learning では, Q 関数をニューラルネットワーク $Q_\theta(s, a)$ で推定する. $Q_\theta(s, a)$ は, 以下の損失関数を繰り返し最小化することで学習される.

$$L_Q(\mathcal{D}) = \mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[(y - Q_\theta(s, a))^2\right]$$
$$y = r + \gamma \max_{a'} Q_{\theta'}(s', a')$$

ここで, $\mathcal{D}$ は状態遷移を保存したリプレイバッファ, s' は状態 s の次の状態である. action repeat を考慮すると, s' は s から 一定の回数だけ行動 a を繰り返して到達する状態である. θ' は固定された古いパラメータセットであり, 一定のステップ間隔で θ と同期される.

学習された Q ネットワーク $Q_\theta(s, a)$ に対し, 方策は以下のように得られる.

$$a = \max_a Q_\theta(s, a)$$

2.3 Soft Actor-Critic

Soft Actor-Critic (SAC) [3], [16] は連続制御タスクで広く用いられているアルゴリズムである. SAC では Q ネットワーク $Q_\theta(s, a)$ に加え, 方策 $\pi_\phi(a|s)$ もニューラルネットワークにより学習する. 方策は正規分布 $\mathcal{N}(\mu_\phi(s), \sigma_\phi(s))$ を tanh で正規化したもので表され, ノイズ $\varepsilon \sim \mathcal{N}(0, I)$ を用いて行動 $a = \tanh(\mu_\phi(s) + \sigma_\phi(s) \odot \varepsilon)$ がサンプルされる.

Q ネットワークは以下の損失関数を最小化することで学習する.

$$L_Q(\mathcal{D}) = \mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[(y_i - Q_\theta(s,a))^2 \right]$$
$$y_i = r + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')} \left[Q_{\theta'}(s',a') - \alpha \log \pi_\theta(a'|s') \right]$$

ここで, α は温度パラメータ, θ' はパラメータ θ の指数移動平均である.

方策は以下の損失関数を最小化することで学習する.

$$L_\pi(\mathcal{D}) = \mathbb{E}_{s \sim \mathcal{D}} \left[D_{\mathrm{KL}}(\pi_\theta(\cdot|s_t) || \exp(\frac{1}{\alpha} Q(s,\cdot))) \right]$$

最後に, 温度パラメータ α はエントロピーが目標値 $\bar{H}$ に近づくように調整される.

$$L_\alpha(\mathcal{D}) = \mathbb{E}_{\substack{s \sim \mathcal{D} \\ a \sim \pi_\theta(\cdot|s)}} \left[-\alpha \log \pi_\theta(a|s) - \alpha \bar{H} \right]$$

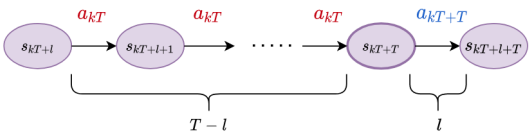
エントロピーの目標値 $\bar{H}$ は $-|\mathcal{A}|$ に設定されることが一般的である.

2.4 action repeat

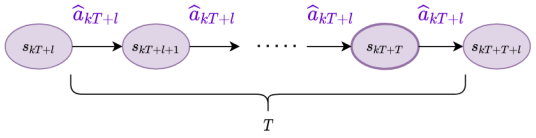
環境から見たステップを環境ステップ, エージェントから見たステップをエージェントステップと呼ぶこととする. action repeat の長さを T とすると, 1 エージェントステップは T 環境ステップに対応し, エージェントは状態 $s_0, s_T, s_{2T}, \ldots$ において行動 $a_0, a_T, a_{2T}, \ldots$ を選択する. それら以外の環境ステップでの行動は, 以前選んだ行動が繰り返される. すなわち, $a_0 = a_1 = \ldots = a_{T-1}$, $a_T = a_{T+1} = \ldots = a_{2T-1}$, $\ldots$ である. 従来の強化学習において, 方策の学習には $s_0, s_T, s_{2T}, \ldots$ のみが用いられ, その間のデータは利用されない.

このように同じ行動を繰り返すのにはいくつかの利点がある. まず, action-gap [17] を増大させ, 安定した学習につながる [12]. これは, 価値推定が不確実な場合に, 行動ごとの違いが明確な方が行動を正しく順位付けしやすいためである. また, 長い action repeat は探索を促進することが期待できる [13]. 例えば迷路を探索するようなゲームの場合, 同じ行動を何度も繰り返して長く進むことで, それまで到達できていなかった状態にたどりつけることがある. さらに, エージェントが行動選択する頻度を減らすことで, 計算コストを削減することができる [13].

action repeat の値 T は通常ハイパーパラメータとして設定され, その値はスコアに大きく影響する. Braylan らの研究によると, Atari 2600 ではタスクごとに適切な action repeat を設定することは非常に重要であり, 例えば Seaquest では 180 という非常に大きな値の場合に最も高い性能が得られた [6]. 連続制御タスクのベンチマークである DeepMind control suite においてもタスクによって高いパフォーマンスを得やすい action repeat が異なる. そのため, タスクごとに異なる action repeat を選んでいる Planet Benchmark [10] よりも, action repeat を固定している Dreamer Benchmark [5] の方が難易度が高いとされている [4].



(a) action repeat 間のデータ



(b) 提案手法で利用される擬似的な遷移データ

図 1: 提案手法の概要. 図は $l = 1$ の場合.

3. 提案手法

第 2 節で述べたように, 一般的な学習では $s_0, s_T, s_{2T}, \ldots$ のみが用いられ, その間のデータは利用されない. 本研究では, このような通常使われないデータを活用して学習の改善を目指す.

3.1 概要

DQN や SAC など多くのオフポリシー強化学習手法では Q 関数のフィッティングが必要である. 提案手法は, Q 関数の学習に action repeat 間のデータを活用する.

行動を繰り返す間のデータについて, 1 エージェントステップ分の遷移は $(s_{kT+l}, a_{kT+l}, s_{kT+l+T})$ と表される. ただし, $k, l \in \mathbb{N}, 0 < l < T$ とする. このデータがそのままでは利用できない理由は, 一般的には, 状態 s_{kT+l} から s_{kT+l+T} まで行動 $a_{kT+l} (= a_{kT})$ が T 回繰り返されるわけではないからである. 図 1a に示すように, エージェントは $t = kT + T$ において新しい行動 a_{kT+T} を選択するため, $a_{kT} = a_{kT+T}$ でない限り s_{kT+l} からは行動 a_{kT} は $T - l$ 回しか繰り返されない.

そこで提案手法では図 1b のように, 状態 s_{kT+l} から擬似的な行動 $\hat{a}_{kT+l}$ が T 回繰り返されたと考える. これにより, 擬似的な遷移データ $(s_{kT+l}, \hat{a}_{kT+l}, s_{kT+l+T})$ を学習に用いることができるようになる.

以降は, 行動空間が連続と離散の場合に分けて詳細に説明する.

3.2 連続行動

擬似的な行動を定めるシンプルな方法として, 行動 $a_{kT+l}, a_{kT+l+1}, \ldots, a_{kT+l+T-1}$ の平均を用いる.

$$\hat{a}_{kT+l} = \frac{1}{T} \sum_{t=0}^{T-1} a_{kT+l+t} \tag{1}$$

環境のダイナミクスが局所的に線形近似できる場合, 行動の平均を用いることによって正規のデータと擬似的なデータを同様に扱うことができる. 詳細は Appendix A.1

で述べる.

3.3 離散行動

行動が離散的な場合, 単純に平均を取ることはできない. そこで, Q ネットワークの構成を変えることにより連続行動と同様に扱えるようにする.

一般的に連続行動, 離散行動それぞれに対する Q 関数は図 2a, 2b のようなネットワークで実装される. 本研究では画像入力を扱うが, 図を簡略化するため畳み込み層は省略した. 図のように, 連続行動では入力ごとに 1 つの Q 値が出力されるのに対し, 離散行動では全行動に関する Q 値が出力される.

離散行動のネットワークを連続行動のネットワークに近づけるため, 図 2c のような構成に変更する. 行動は埋め込み表現に変換され, 1 つの Q 値が出力される. この埋め込み表現は, 目的関数を最適化する中で Q ネットワークのパラメータとともに学習される. これにより, 埋め込み表現の平均を擬似的な行動として用いることができる. つまり, 行動 a の埋め込み表現を $e_\theta(a)$ とすると, 擬似的な行動 $\hat{a}_{kT+l}$ の埋め込み表現 $e_\theta(\hat{a}_{kT+l})$ は

$$e_\theta(\hat{a}_{kT+l}) = \frac{1}{T}\sum_{t=0}^{T-1} e_\theta(a_{kT+l+t})$$

と表される.

3.4 実装

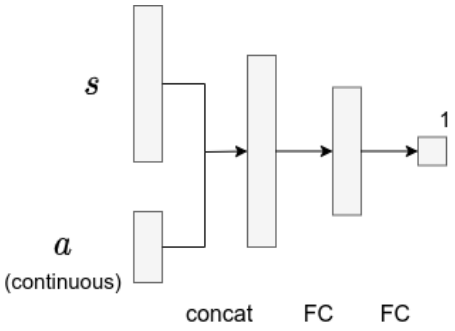
提案手法の変更点は, 正規のデータに追加して擬似的なデータを用いることだけであり, 容易に実装できる. 連続制御タスクにおいてミニバッチを作成する疑似コードを Algorithm 1 に示す. 離散制御タスクでは, 行動の埋め込み表現 $e_\theta(a)$ を Q ネットワークの学習中に獲得し, それらに関して平均を取る.

我々は, 正規のデータと擬似的なデータを平等に扱うことにした. すなわち, サンプルされる遷移データのインデックス i は一様に選ばれる. これにより, 正規データ $(l = 0)$ と疑似的なデータ $(0 < l < T)$ の量の比は $1 : T - 1$ となる. 一方, 正規データのみを使う場合は通常の学習になる. いずれの場合もミニバッチサイズ N は同じ値にした.

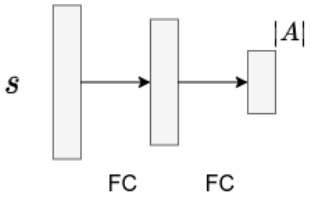
擬似的なデータは近似の上で得られるため, 学習に悪影響を及ぼす可能性も考えられる. そのため, 正規のデータを重視するよう重み付けをすることが有益かもしれない. これは今後の研究課題とする.

4. 実験

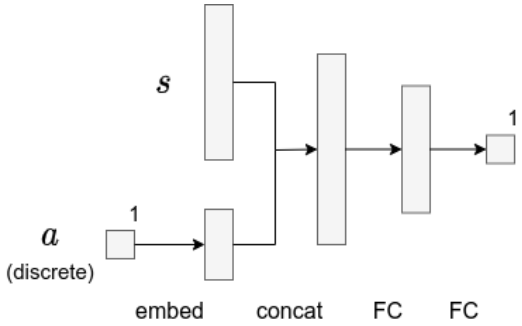
OpenAI Gym のいくつかの環境において, 提案手法を通常の学習と比較する. すべての環境で観測は画像であり, 直近 4 フレームをスタックしたものが状態として用いられた. 学習の環境ステップ数はいずれの場合も 400k に固定した. すべての実験は 5 個のランダムシードで行った. モデルの構成やハイパーパラメータなどの詳細は Appendix A.2, A.3 で述べる.



(a) 連続行動の Q ネットワーク



(b) 離散行動の Q ネットワーク



(c) 提案手法における離散行動の Q ネットワーク

図 2: Q ネットワークの構成図. ”concat”は結合 (concatenate), ”FC”は全結合層 (fully connected), ”embed”は埋め込み (embedding) を表している.

アルゴリズム 1 Q ネットワーク学習用のミニバッチ作成

Input: replay buffer $\mathcal{D}$, mini-batch size N, action repeat parameter T
Output: mini-batch $\mathcal{B}$
  Initialize $\mathcal{B} = \{\}$
  for $k = 1$ to N **do**
    Sample $(s_i, a_i, r_i, \ldots, s_{i+T-1}, a_{i+T-1}, r_{i+T-1}, s_{i+T}) \sim \mathcal{D}$
      where $i = kT + l \quad (k \in \mathbb{N}, l \in [0, T))$
    Current state $s = s_i$
    Next state $s' = s_{i+T}$
    Reward $r = \sum_{t=0}^{T-1} r_{i+t}$
    Pseudo-action $\hat{a} = \frac{1}{T}\sum_{t=0}^{T-1} a_{i+t}$
    Add $(s, \hat{a}, r, s')$ to $\mathcal{B}$
  end for
  return $\mathcal{B}$

4.1 連続制御タスク

連続制御タスクにおける提案手法の有用性を検証するため, Pendulum, CarRacing で実験を行った. 強化学習手法には SAC を用いた. SAC では Q ネットワークに加えて方策ネットワークも学習するが, 比較を平等にするため, 方策ネットワークの学習に action repeat 間のデータは用いていない.

結果を図 3 に示す. action repeat 間のデータを使わないベースラインと比較して, 特に $T = 8$ の場合に提案手法のパフォーマンスが高い結果となった. これは, $T = 8$ のときベースラインは全体の 1/8 のデータしか使うことができないのに対し, 提案手法ではすべてのデータを使うことができるためだと考えられる. さらに Pendulum では, $T = 8$ の場合にベースラインの学習が全く進んでいないことを考慮すると, 8 枚のうち 1 フレームだけでは環境のダイナミクスを捉えることが難しかった可能性もある. 一方, $T = 4$ の場合 Pendulum ではパフォーマンスに違いはなく, CarRacing では提案手法の方が少し良い結果となった. この場合はベースラインで失われるデータ量が先程より少なく, 学習が成功していると考えられる.

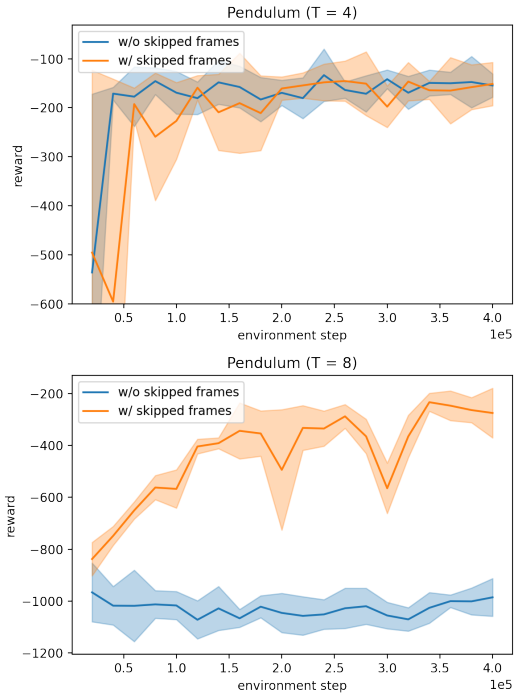
4.2 離散制御タスク

離散制御タスクにおける提案手法の有用性を検証するため, Breakout, Freeway で実験を行った. 強化学習手法には Deep Q-learning を用いた. ただし, 第 3 節で述べたように Q ネットワークの構成は通常とは異なり, 行動の埋め込み表現が使われている. また探索のための手法として, Noisy Network [18] を用いた.

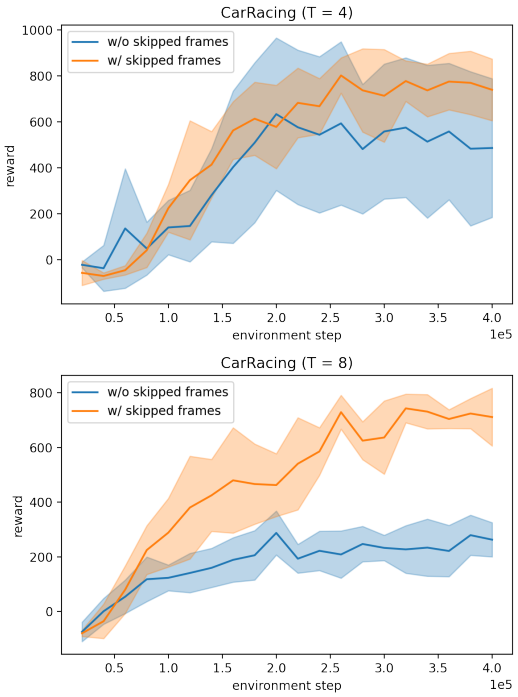
結果を図 4 に示す. Q ネットワークの構成を変えたことによりスコアが著しく低下していないことを確かめるため, 標準的なネットワークを用いた Double DQN [19], Rainbow [1] を同じ環境ステップ数で学習した結果も表示してある. 離散制御タスクでも連続制御タスクと同様に, $T = 4$ の場合は大きな違いがなく, $T = 8$ の場合は提案手法の方が高いパフォーマンスを示した.

4.3 action repeat の効果

$T = 1, 4, 8$ それぞれにおける提案手法の結果を比較し, action repeat の効果を評価する. $T = 1$ の場合, 提案手法は通常の学習と同じになる. 結果を図 5 に示す. Pendulum 以外では, action repeat が学習に一定の良い影響を与えていることがわかる. これは第 2 節で述べたように, action-gap が大きくなったことや探索が促進されたことによると考えられる. このように, action repeat は多くのタスクに必要であり, その上でスキップされたフレームを活用することは重要な意味を持つ.



(a) Pendulum

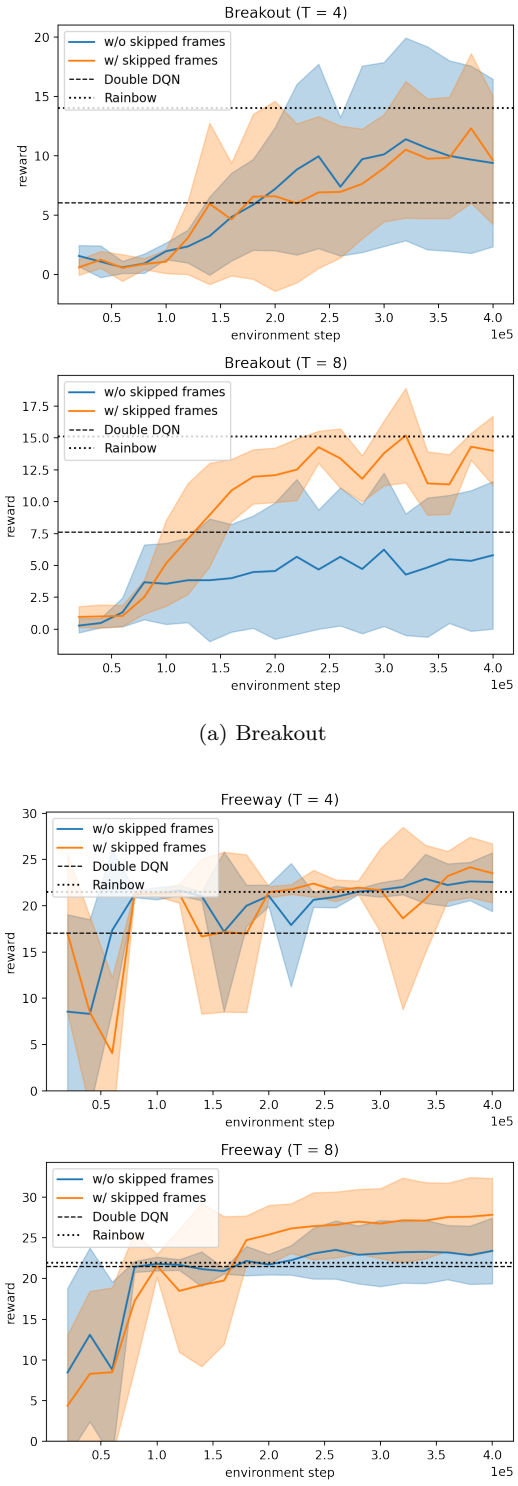


(b) CarRacing

図 3: 連続制御タスクの結果. 実線は平均値, 色付き部分は標準偏差を表している.

4.4 考察

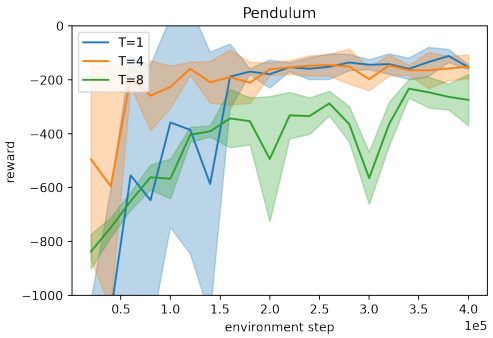
今回の実験では, 行動の連続・離散に関わらず, action repeat が長い場合に提案手法の成績が高く, そうでない場合に提案手法とベースラインは同程度の成績となった. これは自然な結果ではあるが, $T = 4$ の場合も提案手法で活



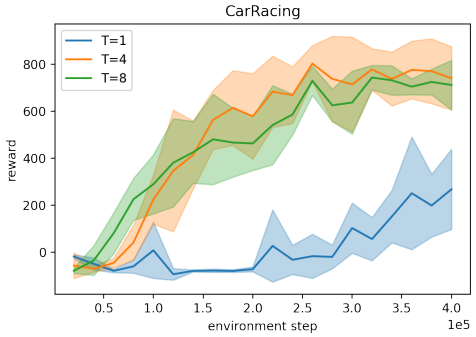
(a) Breakout

(b) Freeway

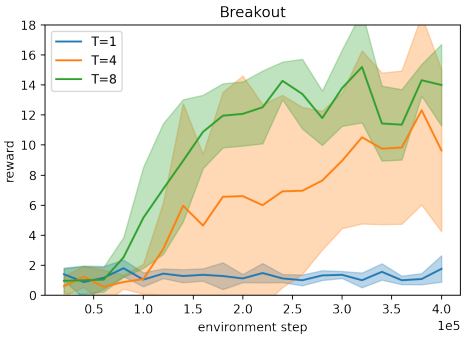
図 4: 離散制御タスクの結果. 実線は平均値, 色付き部分は標準偏差を表している. 破線は Double DQN, Rainbow の最終スコアの平均値を示す.



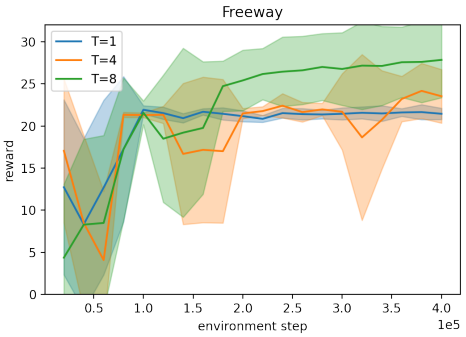
(a) Pendulum



(b) CarRacing



(c) Breakout



(d) Freeway

図 5: 異なる action repeat におけるパフォーマンスの比較. 実線は平均値, 色付き部分は標準偏差を表している.

用できるデータ量はベースラインの 4 倍になっており, より良いパフォーマンスが期待される.

そのような結果が得られなかった原因としては, 提案手法における擬似的な行動のモデリングが最適でなかったため, 追加されたデータが学習に悪影響を及ぼした可能性がある. 今回はシンプルな方法として, 行動の平均により擬

似的な行動を計算したが, 状態の変化から行動を推測する逆モデル [20], [21] などを用いることでより良い表現が得られるかもしれない.

5. おわりに

本研究では, 通常利用されていない action repeat 間のデータを連続・離散両方の制御タスクで活用する手法を提案した. そして OpenAI Gym の環境で実験を行い, その有効性を示した. またいくつかのタスクにおいて, 長い action repeat がパフォーマンスの向上につながることが確認され, 行動を繰り返すことの理論的な利得が実証される結果となった.

今後の課題は, 手法を改善し, より洗練された方法で擬似的な行動を獲得することである. 第 4 節で述べたように提案手法は改善の余地があり, さらにパフォーマンスを向上させられる可能性がある.

本研究ではオンライン強化学習に取り組んだが, エージェントが環境と相互作用せずに学習するオフライン強化学習が近年盛んに研究されている [22]. オフライン強化学習では与えられたデータのみから方策を学習する必要があり, それ以上にデータを集めることはできないため, 提案手法のようにデータを拡張する手法は重要であると考えられる. データ収集時に action repeat 間のデータを保存していれば提案手法はオフライン強化学習にも適用でき, この設定における研究も興味深いと思われる.

参考文献

[1] Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. and Silver, D.: Rainbow: Combining Improvements in Deep Reinforcement Learning, *AAAI Conference on Artificial Intelligence* (2018).

[2] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O.: Proximal Policy Optimization Algorithms, *arXiv:1707.06347* (2017).

[3] Haarnoja, T., Zhou, A., Abbeel, P. and Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, *International Conference on Machine Learning*, pp. 1861–1870 (2018).

[4] Kostrikov, I., Yarats, D. and Fergus, R.: Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels, *arXiv:2004.13649* (2020).

[5] Hafner, D., Lillicrap, T., Ba, J. and Norouzi, M.: Dream to Control: Learning Behaviors by Latent Imagination, *International Conference on Learning Representations* (2020).

[6] Braylan, A., Hollenbeck, M., Meyerson, E. and Miikkulainen, R.: Frame Skip Is a Powerful Parameter for Learning to Play Atari, *AAAI-15 Workshop on Learning for General Competency in Video Games*, p. 2 (2015).

[7] Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M. and Bowling, M.: Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents, *Journal of Artificial Intelligence Research* (2017).

[8] Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G. and Michalewski, H.: Model-Based Reinforcement Learning for Atari, *International Conference on Learning Representations* (2020).

[9] Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T. and Riedmiller, M.: DeepMind Control Suite, *arXiv:1801.00690* (2018).

[10] Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H. and Davidson, J.: Learning Latent Dynamics for Planning from Pixels, *International Conference on Machine Learning*, pp. 2555–2565 (2019).

[11] Lee, A. X., Nagabandi, A., Abbeel, P. and Levine, S.: Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model, *arXiv:1907.00953* (2019).

[12] Bellemare, M. G., Ostrovski, G., Guez, A., Thomas, P. S. and Munos, R.: Increasing the Action Gap: New Operators for Reinforcement Learning, *AAAI Conference on Artificial Intelligence* (2016).

[13] Hessel, M., van Hasselt, H., Modayil, J. and Silver, D.: On Inductive Biases in Deep Reinforcement Learning, *International Conference on Learning Representations* (2019).

[14] Bellman, R.: A Markovian Decision Process, *Indiana University Mathematics Journal*, pp. 679–684 (1957).

[15] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M.: Playing Atari with Deep Reinforcement Learning, *NIPS Deep Learning Workshop* (2013).

[16] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P. and Levine, S.: Soft Actor-Critic Algorithms and Applications, *arXiv:1812.05905* (2019).

[17] Farahmand, A.-m.: Action-Gap Phenomenon in Reinforcement Learning, *Advances in Neural Information Processing Systems*, pp. 172–180 (2011).

[18] Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C. and Legg, S.: Noisy Networks for Exploration, *International Conference on Learning Representations* (2018).

[19] van Hasselt, H., Guez, A. and Silver, D.: Deep Reinforcement Learning with Double Q-Learning, *AAAI Conference on Artificial Intelligence*, pp. 2094–2100 (2016).

[20] Pathak, D., Agrawal, P., Efros, A. A. and Darrell, T.: Curiosity-Driven Exploration by Self-Supervised Prediction, *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 488–489 (2017).

[21] Chandak, Y., Theocharous, G., Kostas, J., Jordan, S. and Thomas, P. S.: Learning Action Representations for Reinforcement Learning, *arXiv:1902.00183* (2019).

[22] Levine, S., Kumar, A., Tucker, G. and Fu, J.: Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems, *arXiv:2005.01643* (2020).

[23] Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J. and Fergus, R.: Improving Sample Efficiency in Model-Free Reinforcement Learning from Images, *arXiv:1910.01741* (2020).

[24] Srinivas, A., Laskin, M. and Abbeel, P.: CURL: Con-

trastive Unsupervised Representations for Reinforcement Learning, *International Conference on Machine Learning*, pp. 10360–10371 (2020).
[25] Ba, J. L., Kiros, J. R. and Hinton, G. E.: Layer Normalization, *arXiv:1607.06450* (2016).
[26] Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S. and van Hasselt, H.: Multi-Task Deep Reinforcement Learning with PopArt, *arXiv:1809.04474* (2018).

付　　録

A.1　擬似的な行動に関する詳細

第 3 節で述べたとおり, action repeat 間のデータに関しては 状態 s_{kT+l} から行動 $a_{kT+l}(= a_{kT})$ を T 回繰り返して状態 s_{kT+l+T} に至るわけではない．そのため, そのままでは正規のデータと同様に扱うことはできない．しかし, 行動の平均として得られる擬似的な行動 $\hat{a}_{kT+l}$ に関しては, それを s_{kT+l} から T 回繰り返すと s_{kT+l+T} に近似的に等しい状態に至る．

ここでは, 連続時間の設定で考える．エージェントの状態 $x(t)$, 行動 $u(t)$ が何らかのダイナミクス f を用いて

$$x(t+T) = \int_t^{t+T} f(x(t'), u(t'))\, dt'$$

という関係にあるとする．状態が $t \le t' \le t+T$ において急激に変化しないと仮定すると,

$$x(t+T) \approx \int_t^{t+T} f(x(t), u(t'))\, dt'$$

と近似できる．ここで,

$$u(t) = \begin{cases} u_1 & (t \le t + pT) \\ u_2 & (t > t + pT) \end{cases}$$

とする．ただし, p は $0 < p < 1$ を満たす実数である．すなわち, エージェントは時刻 $t+pT$ までは行動 u_1 を繰り返し, その後は新しい行動 u_2 を繰り返す.

$$\begin{aligned} x(t+T) &\approx \int_t^{t+pT} f(x(t), u_1)\, dt' + \int_{t+pT}^{t+T} f(x(t), u_2)\, dt' \\ &= pT\, f(x(t), u_1) + (1-p)T\, f(x(t), u_2) \end{aligned}$$

ここで, $\hat{u} = p\, u_1 + (1-p)\, u_2$ を行動の平均とする.

$$u_1 = \hat{u} - (1-p)\,(u_1 - u_2), \quad u_2 = \hat{u} + p\,(u_1 - u_2)$$

と表されることに注意し, 行動に関して 1 次近似すると,

$$\begin{aligned} x(t+T) &\approx pT\, f(x(t), \hat{u} - (1-p)\,(u_1 - u_2)) \\ &\quad + (1-p)T\, f(x(t), \hat{u} + p\,(u_1 - u_2)) \\ &\approx pT \left\{ f(x(t), \hat{u}) - (1-p)\,(u_1 - u_2)\, \frac{\partial f}{\partial u}(x(t), \hat{u}) \right\} \\ &\quad + (1-p)T \left\{ f(x(t), \hat{u}) + p\,(u_1 - u_2)\, \frac{\partial f}{\partial u}(x(t), \hat{u}) \right\} \\ &= T f(x(t), \hat{u}) \\ &= \int_t^{t+T} f(x(t), \hat{u})\, dt' \approx \int_t^{t+T} f(x(t'), \hat{u})\, dt' \end{aligned}$$

つまり, $u(t)$ を定数 $\hat{u}$ に置き換えても, 次の状態 $x(t+T)$ は概ね変化しない.

A.2　ニューラルネットワークの構成

第 3 節で述べたとおり, 我々の手法は離散制御タスクのネットワークを連続制御タスクのものに近づける．そのため, 画像をエンコードするネットワークと Q ネットワークは同じ構造とした.

A.2.1　エンコーダネットワーク

エンコーダネットワークの構造は SAC+AE [23] を参考にした．このモデルは画像入力から SAC により方策を学習するいくつかの研究で用いられている [4], [24]．このエンコーダは 3×3 のカーネル, 32 のチャネルを持つ畳み込み層 4 層で構成されている．活性化関数には ReLU を用いた．第 1 層のストライドは 2, それ以降の層のストライドは 1 とした．畳み込み層の出力は LayerNorm [25] により正規化され, 全結合層で 50 次元のベクトルに変換された後, tanh が適用される.

A.2.2　Q ネットワークと方策ネットワーク

Q ネットワークと方策ネットワークはいずれも 3 層の全結合層で構成されており, 隠れ層の次元は 256 とした．活性化関数には ReLU を用いた．Q ネットワークはエンコーダネットワークの出力と行動 (または行動の埋め込み表現) を受け取り, 1 つの Q 値を出力する．離散行動の埋め込み表現はすべてのタスクで 8 次元とした．方策ネットワークは連続制御タスクにのみ用いられ, エンコーダネットワークの出力を受け取り, 方策を表す正規分布の平均と分散を出力する.

A.3　その他のハイパーパラメータ

表 A·1 にハイパーパラメータの設定を示す．離散制御タスクでは, 学習を安定させるため Double DQN [19] を用いた．報酬のスケールの違いに対応するため, 連続制御タスクでは PopArt [26] を用い, 離散制御タスクでは $[-1, 1]$ に報酬をクリップした.

表 A·1: 実験に用いたハイパーパラメータ

	連続行動タスク	離散行動タスク
アルゴリズム	SAC	Double DQN
環境ステップ数	400k	400k
画像サイズ	84×84	84×84
フレームスタック	4	4
1 アップデート当たりの環境ステップ	4	4
オプティマイザ	Adam	Adam
学習率	0.001	0.0003
環境ステップ当たりの割引率 γ	$0.99^{1/4}$	$0.99^{1/4}$
ミニバッチサイズ N	32	64
リプレイバッファサイズ	制限なし	制限なし
学習を開始するリプレイバッファサイズ	500	500
エピソード当たりの最大環境ステップ数	制限なし	108k