

複数の入出力サイズを扱う継続的強化学習手法

川島 丸生^{1,a)} 伊庭 斉志¹

概要: 多くの機械学習モデルは、過去のタスクで学習した知識を忘れること無く、新しいタスクを学習することができない。継続的学習は、この問題を解決することを目的とした手法で、1つのモデルが複数のタスクを逐次的に学習する。継続的学習の中でも、ゲームやロボットの操作といった強化学習のタスクを扱う継続的強化学習がある。継続的強化学習手法は入出力を統一するのが一般的であるが、画像入力と状態入力を一緒に扱うことができないようにタスクの範囲が限られている問題がある。そこで本研究では、入出力サイズの異なるタスクを扱える継続的強化学習手法を提案する。提案手法は、継続的学習手法である Learn-to-Grow を拡張することにより、強化学習のアルゴリズムである DDQN と組み合わせた。本稿では、OpenAI Gym Atari のいくつかのタスクで提案手法の有効性を検証し、破滅的忘却を防げていることなどを確認した。

A Continual Reinforcement Learning Method Handling Multiple Input and Output Sizes

MARUKI KAWASHIMA^{1,a)} HITOSHI IBA¹

Abstract: Many machine learning models are unable to learn a new task without forgetting the knowledge learned in past tasks. Continual learning is a method aimed at solving this problem, where one model learns multiple tasks sequentially. Continuous reinforcement learning deals with reinforcement learning tasks such as games and robot operations. Although continual reinforcement learning methods generally unify input and output, there is a problem that the scope of the task is limited such that image and state inputs cannot be handled together. In this study, we propose a continuous reinforcement learning method that can handle tasks with different input and output sizes. The proposed method is an extension of a continuous learning method, Learn-to-Grow, and is combined with a reinforcement learning algorithm, DDQN. In this paper, we tested the effectiveness of the proposed method on several tasks of OpenAI Gym Atari and confirmed that the method prevented catastrophic forgetting.

1. はじめに

人間は、新しいタスクを解き、新しいスキルを習得しても、基本的に既に習得したスキルを失うことはないため、継続的に学習することができる。さらに、新しいタスクに対して、関連性の高いタスクを解いた時に得た知識を用いて効率的に対処することができる。一方で、一般的な Deep Neural Network (DNN) を用いた機械学習では、破滅的忘

却 (catastrophic forgetting) [1] の問題により、新しいタスクについて学習すると、以前に学習した知識をほとんど忘却する [2] ため、このような学習をすることはできない。この問題に対処し、以前の経験を活用して学習方法を学習して、効率的に新しいタスクを学習する手法として、継続的学習 (continual learning) がある [3]。特に、機械学習の中で、より人間の扱うものに近いタスクを扱うことのできる強化学習と、継続的学習を組み合わせた継続的強化学習 (continual reinforcement learning) [4] は、人間のような学習を目指す上で人工知能分野の発展に有用である。

しかし、継続的学習の多くは教師あり学習を扱うことが多く、強化学習に用いることのできる手法は少ない。これ

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)} maruki@g.ecc.u-tokyo.ac.jp

は、強化学習はエージェントが環境にアクセスしてデータを集めなくてはならないため、多くの場合データが与えられる教師あり学習に比べて学習が難しいためである。強化学習に用いることのできる手法であっても、シミュレートされた同種のロボットの様々な実行速度を様々なタスクとして使用する、統一されたサイズの入出力で扱えるタスクのみからなるタスク集合を使用する [6] などの限定的なタスク集合に対しての手法が多い。このような狭いタスク集合を扱う継続的学習では、完全に新しいタスクを扱うことは難しい。また、異なる入出力サイズを扱うことができないため、同じロボットやゲームで状態入力による学習の後に、画像入力の学習を効率的に行うといった転移学習も行うことができない。そのため、より広範なタスクの効率的な学習を目指す上で、異なる入出力サイズを含む広いタスク集合を扱える継続的強化学習手法が必要となる。

そこで本研究では、より広範なタスクの効率的な学習を目指すために、知識の共有が難しいタスクや求められる入出力の形式が異なるタスクを含めたタスク集合に対して、継続的強化学習を行うことを目的とする。

本稿では、画像入力と状態入力の異なる入力形式を扱えるモデルを提案する。いくつかの連続タスクで破滅的忘却が防げて、学習済みのパラメータを有効に使えることを確認した。また、全結合層に少量のパラメータを追加する場合のモデルを作成し、Learn to Grow [5] の手法を全結合層を含むモデルに適用できるよう拡張した。

2. 関連研究

2.1 強化学習

強化学習は、機械学習の手法の一つで、エージェントが環境から観測した状態を元に決定し、行動によって環境から報酬を受け取り、受け取る報酬の総和が最大になるような方を学習するというものである。強化学習以外の機械学習の代表的な手法には、教師あり学習と教師なし学習がある。この2つの手法は、どちらも環境との相互作用のようなものがなく、一方的に入力されるデータに対する分析手法であるのに対して、強化学習はデータを集める部分も含めて決定する手法である。

2.1.1 Q 学習

Q 学習 [7] は、強化学習の代表的な手法の一つである。Q 学習では、環境を有限マルコフ決定過程 (Markov decision process, MDP) でモデル化できると仮定する。MDP は、次の4要素の組 $\langle S, A, T, R \rangle$ で表されるモデルである。

- $S = \{s_1, s_2, \dots, s_N\}$: 状態 (state) の有限集合
- $A = \{a_1, a_2, \dots, a_M\}$: 行動 (action) の有限集合
- $T: S \times A \times S \rightarrow [0, 1]$: 遷移 (transition) 関数
- $S \times A \rightarrow \mathbb{R}$: 報酬 (reward) 関数

遷移関数 T は、ある状態 $s \in S$ で行動 $a \in A$ を選択して状態 $s' \in S$ に遷移する確率である。また、報酬関数 R は、状

態 s で行動 a を選択した時の報酬である。MDP では、次の可能な行動は現在の状態のみによって決まるので、現在の状態 s が与えられた時に次の行動 a を決定する方策は、状態 s で行動 a を取る確率 $\pi(s, a)$ と表される。最大化する目的関数である、ある状態 s で行動 a を選択した時に受け取る報酬の総和は以下の様に表される。

$$Q^\pi(s, a) = E \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) | s_0 = s \right] \quad (1)$$

ここで、 γ ($0 < \gamma \leq 1$) は割引率と呼ばれる定数で、時間的に近い報酬を重視することで不要な行動を減らし、行動にかかる時間が非常に長くなるのを避けるための値で、多くの場合は $0.9 \leq \gamma \leq 0.99$ のような1に近い値に設定される。この目的関数は、最適化された方策を π^* とすると、ベルマン方程式 (Bellman equation) と呼ばれる

$$Q^{\pi^*}(s, a) = R(s, \pi^*(s)) + \gamma \max_{s'} \sum_{s'} T(s, a, s') Q^{\pi^*}(s', \pi^*(s')) \quad (2)$$

のような形で表され、この行動価値関数 $Q^{\pi^*}(s, a)$ を、関数 $Q(s, a)$ で近似することで学習を行うのが Q 学習であり、次の式で更新を行う。

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \left(R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \quad (3)$$

ここで、 α_t は学習率と呼ばれる値で、次の2つの条件を満たす時、関数 $Q(s, a)$ は必ず $Q^{\pi^*}(s, a)$ に収束することが証明されている。多くの場合、学習率は0.1程度に設定される。

$$\sum_{t=0}^{\infty} \alpha_t \rightarrow \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty \quad (4)$$

2.1.2 Deep Q-Network (DQN)

Deep Q-Network (DQN) [8] は、Q 学習にニューラルネットワークを用いて行動価値関数 $Q(s, a)$ を近似する手法である。DQN では、ニューラルネットワークのパラメータを θ として、行動価値関数を $Q(s, a; \theta)$ と表現し、次の誤差関数を最小化するように学習を行う。

$$L_t(\theta) = E \left[\left(R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a; \theta) - Q(s_t, a_t; \theta) \right)^2 \right] \quad (5)$$

この誤差関数を微分して得られる次の式を用いて誤差逆伝播 [9] を行うことで、ニューラルネットワークの学習を行う。

$$\nabla_{\theta} L(\theta) = E \left[\left(R(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a; \theta) - Q(s_t, a_t; \theta) \right) \nabla_{\theta} Q(s_t, a_t; \theta) \right] \quad (6)$$

2.1.3 Double Deep Q-Network (DDQN)

DQN では、Q の評価に誤差が含まれるため、価値の高い行動を過大評価してしまう問題がある。この問題に対処するため、Hasselt らは Double Deep Q-Network (DDQN) [10] と呼ばれる手法を提案した。DDQN は、行動を選択するための Q 関数と行動を評価する Q 関数を分けることで、行動の過大評価を抑制した。DDQN の誤差関数は、以下のようになる。

$$L_t(\theta) = E \left[\left(R(s_t, a_t) + \gamma Q(s_{t+1}, \arg \max_a Q(s_{t+1}, a; \theta'); \theta) - Q(s_t, a_t; \theta) \right)^2 \right] \quad (7)$$

DQN には、DDQN 以外にも各種問題を解決するために、Multi-step learning や優先度付き経験再生 (Prioritized Experience Replay) [11] などの様々な工夫が提案されている。

2.2 破滅的忘却

破滅的忘却 (catastrophic forgetting) [1] とは、ニューラルネットワークにおいて既に学習したタスクとは異なる新しいタスクを学習したときに、以前のタスクに関する情報を失い、既に学習したタスクに対する精度が低下してしまう問題である。

2.3 継続的学習

継続的学習 (continual learning) とは、破滅的忘却を防ぎながら、複数のタスクを逐次的に学習する手法である。

また、同じく複数のタスクを扱う手法にメタ学習 (meta-learning) がある。メタ学習とは、学習方法を学習する (learning to learn) というもので、タスク集合の各タスクの学習の隠れた共通点等を学習することで、タスク集合の全てのタスクに対して効率的な学習を可能にする手法である。

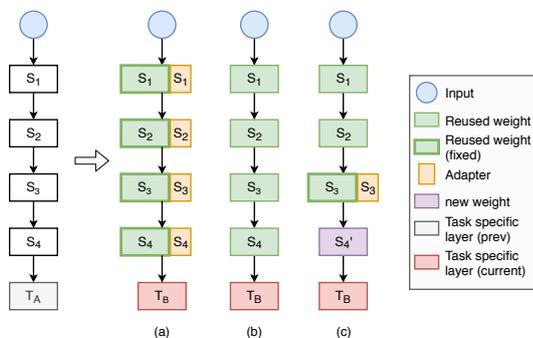


図 1 各学習手法でのモデルのパラメータ変化の様子

図 1 に、いくつか継続的学習やメタ学習の手法において、タスク集合から各タスクの学習を行う途中で、タスク A の

学習の後異なるタスク B の学習をした時の、モデルのパラメータの変化の仕方について示す。

図 1 の (a) は、タスク A までのパラメータ θ_{t-1} を固定して、新たなパラメータ θ_t を並列に導入することでタスク B までのパラメータ Θ_t を学習するものである。Rusu らの Progressive Neural Networks [12] がこの方法に近い。タスク毎に新たに追加するパラメータの導入方法は、多くの場合経験則によるものである。この方法では、過去のパラメータを固定するため破滅的忘却を完全に防止しながら、新しいタスクの学習に過去の学習の知識を用いることができる。増やすパラメータ数を個別にタスクを学習させる時のモデルのパラメータ数と同じにすれば、各タスク毎に個別のモデルで学習させるのとほとんど変わらないため、高い精度ができるが計算コストが非常に高くなるという問題が生じ、増やすパラメータ数を少なくすると、計算コストは低くなるが精度が低下するように、計算効率と精度性能はトレードオフの関係にある。

図 1 の (b) は、タスク毎のパラメータ数を変化させずに学習するものである。この方法は、新しいタスクの学習の際に、単純に勾配法を用いて更新をすると破滅的忘却の問題が生じるため、Kirkpatrick らの Elastic Weight Consolidation (EWC), Zenke らの Synaptic Intelligence [13], Finn らの Model-Agnostic Meta-Learning (MAML) [14] や一次近似により MAML より計算量を減らした Nichol らの Reptile [15] 等のように特殊な制約等を用いてパラメータ変化を制御する必要がある。これらの方法は、タスク集合に対して十分なパラメータ数とモデルであれば良い性能ができるが、多くの場合パラメータ空間の制約により性能は各タスクを個別のモデルで学習した場合に対して劣化する問題がある。また、タスク数が増大したときのパラメータ制御の有効性は十分に調べられていない。

図 1 の (c) は、Li らの Learn to Grow で、新しいタスクを学習する時に、各層のパラメータをそのまま利用するか、新しいパラメータを並列に導入して利用するか、同じ大きさの新しいパラメータにするかをそれぞれ選択するものである。この方法は、図の (a), (b) の両方の利点を活用でき、新しいタスクの計算効率と精度性能を犠牲にすること無く、古いタスクの破滅的忘却を完全に回避することができる。

2.3.1 Learn to Grow

Learn to Grow [5] は、継続的学習手法の一つで、Neural Architecture Search (NAS) の手法の一つである Differentiable Architecture Search (DARTS) [16] と L2 正則化のようなパラメータ最適化手法を組み合わせたものである。論文では、MNIST のような画像入力の教師あり学習において性能を検証している。

Learn to Grow の学習は、図 2 のようにネットワークのレイヤー毎の選択を決定するための構造探索と、レイヤー毎に選択された選択肢のパラメータを更新するパラメータ

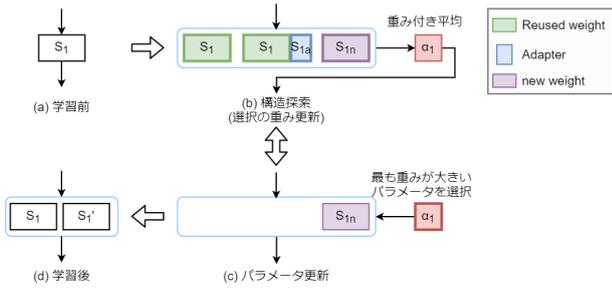


図 2 Learn to Grow のある一層の変化の様子

更新の 2 つに分けられ、最終的に決定された選択枝のパラメータが新しいものであれば追加して保存される。

まず、新たなタスク T_k を学習するにはレイヤー毎に、再利用 (reuse), 小さな重みを追加 (adaptation), 新しく生成 (new) した重みを作成する。モデルの全ての重みを含むスーパーネットワーク S の l レイヤーの過去のタスクで学習済みのパラメータの総数を $|S^l|$ と表すと、ネットワークのレイヤー l の選択枝の総数は $C_l = 2|S^l| + 1$ となる。

構造探索時、各レイヤーの出力は、レイヤーの全ての選択枝の演算結果の重み付き和である。各選択枝の出力 g_c^l は入力を x_l とすると

$$g_c^l(x_l) = \begin{cases} S_c^l(x_l) & (c \leq |S^l|) \\ S_c^l(x_l) + \gamma_{c-|S^l|}^l(x_l) & (|S^l| < c \leq 2|S^l|) \\ o^l(x_l) & (c = 2|S^l| + 1) \end{cases} \quad (8)$$

と表される。ここで、 $\gamma_{c-|S^l|}^l$ は adapt により追加された重みによる演算であり、 o^l は new により追加された重みによる演算である。レイヤー l の c 番目の選択枝の重みを α_c^l とすると、構造探索時のレイヤー l の出力は

$$x_{l+1} = \sum_{c=1}^{C_l} \frac{\exp(\alpha_c^l)}{\sum_{c'=1}^{C_l} \exp(\alpha_{c'}^l)} g_c^l(x_l) \quad (9)$$

と表される。DARTS と同様に、レイヤー毎の選択を全ての選択枝の出力のソフトマックスに緩和することにより、どの重みを選択するかという不連続的な探索空間から、選択枝の重みを調整するという連続的な探索空間にすることで構造探索の学習も勾配法を用いて行えるような工夫が成されている。

パラメータ更新時、レイヤー l の出力は $c_l = \arg \max_c \alpha_c^l$ により選択された選択枝の出力で

$$x_{l+1} = g_{c_l}^l(x_l) \quad (10)$$

と表される。

構造探索では各選択枝の重みである α_i^l のみ学習し、パラメータ更新時に選択された経路の選択枝のパラメータの学習を行う。これを繰り返すことにより、既にスーパーネットワークが保持しているパラメータに対してタスク T_k を学習する際に必要なパラメータが必要な分だけ追加される

ため、効率的な学習を行うことができる。

Learn to Grow では、タスク T_t の学習に用いるパラメータ Θ_t のうち経路が決定されパラメータ更新時のネットワークに使われるのは一部であるため、タスク毎に用いられるパラメータを選択する関数を $s_t(\Theta_t)$ とすると、損失関数は

$$L_t(s_t(\Theta_t)) = \frac{1}{n_t} \sum_{i=1}^{n_t} l_t(f(x_i^{(t)}; s_t(\Theta_t)), y_i^{(t)}) + \beta_t R_t^s(s_t) + \lambda_t R_t^p(\Theta_t) \quad (11)$$

と表される。ここで R_t^s は、構造探索において new の重みの選択ばかりが行われることによって、タスク毎に個別のモデルで学習させるのと変わらない状態になるのを避けるために、パラメータ数の増加を抑えるものであり、 β_t はどの程度パラメータの増加を押さえるかを決定する係数である。レイヤー l の c 番目の選択枝の新しく追加されたパラメータのサイズ (の対数) を z_c^l とすると $R_t^s(s_t) = \sum_{c,l} \alpha_c^l z_c^l$ と表される。 R_t^p は L2 正則化等のパラメータが大きくなりすぎないようにするためのものであり、 λ_t はどの程度パラメータの大きさを押さえるかの係数である。

3. 提案手法

3.1 概要

本研究では、図 3 のように入力部分を画像や状態のような形式やサイズの異なるタスク毎に分けて、中間部分を共有し、出力層を出力サイズ毎に分けたモデルを基本として、Li らの Learn to Grow (LG) [5] を元にパラメータの追加、選択、更新を行うことを提案する。モデルは、Multi-step learning や Prioritized Experience Replay を使用した DDQN の Q ネットワークとして用いる。

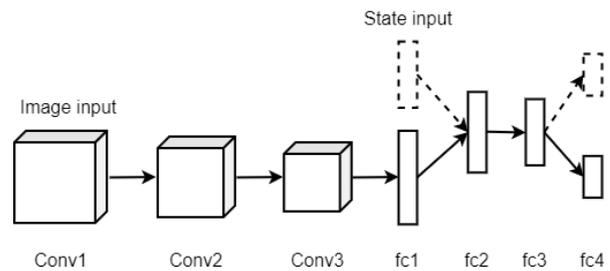


図 3 提案手法のモデル (実線は画像入力の時の経路の例)

学習中の LG では、レイヤーの出力が学習中のタスクで扱う選択枝の出力のソフトマックスとなる構造探索と、選択枝の重み α が最も大きいパラメータの出力となるパラメータ更新がある。提案手法では、選択されている経路の評価が行えるように、エージェントが環境に対して行動を決定する時にはレイヤーの出力を選択枝の重みが最も大きいパラメータの出力とした。

モデルやパラメータの更新は、通常の DDQN における

Q ネットワークの更新時に行う。選択肢の重み α は、途中で固定することでモデルの経路が変化し続けて学習が不安定にならないようにした。

3.2 Learn to Grow の拡張

LG は、タスク毎に出力層のパラメータを変えていたが、入力形式を合わせた画像入力であったため、提案手法では入力部分もタスク毎にパラメータを変えられるようにした。

また、LG は畳み込みニューラルネットワーク (Convolutional Neural Network, CNN) に適用されていたため、adapt の演算は畳み込み層である reuse の出力と出力にカーネルサイズ 1 の畳み込みの出力を足し合わせるもので、全結合層には用いることができない。そのため、提案手法における全結合層の adapt は、同様に少量のパラメータで出力を少し変化させるように、2 層の全結合層によりレイヤーの入力を一度小さくしてから出力したものを reuse の出力に足し合わせる演算とした。

4. 実験

4.1 実験環境

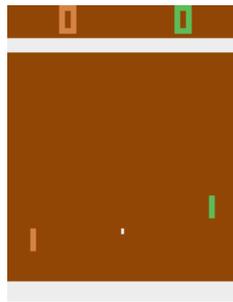


図 4 Freeway のプレイ画面 図 5 Pong のプレイ画面

提案手法の有効性を検証するために、同じゲームで画像入力と状態 (ram) 入力の 2 通りの入力が可能で OpenAI Gym の Atari で、DDQN でも学習が容易な Freeway と Pong という 2 つのゲームを元に連続タスクを作成し、学習を行った。

画像入力時には、 84×84 のグレースケール画像に変換し 4 フレームをまとめて $4 \times 84 \times 84$ を 1 つの状態として学習を行った。

また、状態入力は Atari の 128 バイトの内部メモリを 128 個の uint8 型の配列として入力するもので、フレームカウンタやスコア、プレイヤー等のオブジェクトの位置や速度の情報を含んでいる。

4.1.1 Freeway

Freeway は図 4 のように、ニワトリが道路を横断するゲームである。ニワトリは、止まるか前に進むか後ろに進むことができ、道路を横断すると 1 点を獲得して手前に戻る。ニワトリが道路の車に当たると、約 2 車線分手前に戻される。

報酬は横断した時の 1 のみで他は 0 であり、ゲームは 8192 フレームで終了する。

常に前に進む行動を取り続けた場合の得点は 23 点程度であり、車に当たらないような行動をすると 30 点を超えることができる。

4.1.2 Pong

Pong は図 5 のように、右側のパドルを上下に操作してボールを左側の CPU と打ち合う卓球ゲームである。相手がボールを打ち返すことができなければ自分が 1 点、自分がボールを打ち返すことができなければ相手が 1 点獲得し、どちらかが 21 点を取るとゲームは終了する。報酬は自分が得点した時に 1、得点された時に -1 で他は 0 である。

4.2 実験方法

提案手法が、破滅的忘却を防いでいるか、学習したパラメータを他のタスクや入力形式の異なるタスクの学習に利用できているかを調べるために 1 番目と 3 番目のタスク (task1, 3) を同じにした 3 連続タスクで実験 1-3 を行った。比較対象として、提案手法の基本モデルと同じ構造の Q ネットワークを用いた DDQN エージェントを利用した。

モデルは、画像入力用の入力部分である畳み込み層 3 つ conv1-3 と全結合層 fc1、状態入力用の入力部分である全結合層 fc1、共通部分である全結合層 fc2-3、出力層である全結合層 fc4 の 7 層を基本に構成した。ハイパーパラメータ等の詳細は付録 A.1, A.2 で述べる。

表 1 各実験の task1, task2(task3 は task1 と同じ)

	task1	task2
実験 1	Freeway	Freeway-ram
実験 2	Freeway-ram	Freeway-ram(行動変形)
実験 3	Freeway	Pong

各実験のタスク設定を表 1 に示す。実験 1 では、異なる入力形式を含む連続タスクで有効かを確認するために、同じゲームの画像入力と状態入力を学習する。比較対象では、task2 の学習開始時には task1 で学習した fc2-4 のパラメータを、task3 の学習開始時には task1 で学習した conv1-3, fc1 と task2 で学習した fc2-4 のパラメータを用いる。

実験 2 では、出力層だけ適切にパラメータを増やすことができるかを確認するために、ネットワークの出力に対応する行動の順番を入れ替えて学習する。task2 では行動の変形により、通常での (停止, 前進, 後退) の各行動が (前進, 後退, 停止) となる。

実験 3 では、異なるタスクを含む連続タスクで有効かを確認するために、2 つのゲームを学習する。比較対象では、task2 の学習開始時には task1 で学習した conv1-3, fc1-3 のパラメータを、task3 の学習開始時には task2 で学習した conv1-3, fc1-3 と task1 で学習した fc4 のパラメータを用いる。

4.3 実験結果

実験はそれぞれ5個のランダムシードで行った。実験結果の図の実線は中央値、色付き部分は最小値から最大値を表している。図中では提案手法をLGDDQN、比較手法をDDQNとしており、連続タスクの中で一番始めのタスクであるtask1ではどちらの手法も全く同じなので表記していない。

4.3.1 実験 1: Freeway 画像-状態-画像

結果を図6, 7, 8に示す。図6, 8より、提案手法ではtask2の学習を挟んでもすぐに30以上の報酬を得ており破滅的忘却を防いでいる。一方で、既存手法ではtask2の状態を挟んだことにより、破滅的忘却が生じてtask1に近い学習になっている。task1よりも学習が早いのは、task2で更新されたパラメータはfc2-4であり、画像入力で用いるconv1-3とfc1のパラメータは保存されていたこと、task2はtask1と入力形式が異なる同じゲームなのでfc2-4でもあまり忘却が生じなかったためと考えられる。

図7では、ばらつきが大きいものの提案手法の方が学習が速い。提案手法のモデルのfc2-3層では、task1で学習したパラメータを再利用(reuse)したものが多く、入力が異なっても同じゲームであるため、task1で学習したパラメータを有効に使うことができている。

4.3.2 実験 2: Freeway 状態-状態(行動変形)-状態

結果を図9, 10, 11に示す。図9, 11より既存手法と提案手法のどちらもtask1の最終性能付近にも比較的早く到達しており破滅的忘却はあまり生じなかった。また、図10, 11より提案手法と既存手法の学習結果はほとんど同じである。これは、task1とtask2がほぼ同じであったため、パラメータをあまり更新する必要がなく、破滅的忘却もあまり生じず差が生まれなかったと考えられる。

提案手法がtask2で選択したパラメータは、fc2-3はほとんどがreuse、fc4はadaptであり、出力に変化を加えたタスクにおいて出力層のパラメータだけを追加しているため、適切な層へパラメータを追加できている。

4.3.3 実験 3: Freeway 画像-Pong 画像-Freeway 画像

結果を図6, 12, 13に示す。図6, 13より、提案手法ではすぐに30以上の報酬を得ており、破滅的忘却を防いでいる。一方で、既存手法ではtask1の学習結果とほとんど同じになっており、破滅的忘却が生じている。

図12では、提案手法はばらつきが大きい既存手法と似た学習結果になっている。これは、提案手法がtask2で選択したパラメータはほとんどがnewであり、FreewayとPongが比較的似ていないタスクであるために、task1で学習したパラメータを利用することができなかったためである。

4.3.4 考察

実験2では破滅的忘却がほとんど生じず、図8, 13より既存手法ではtask1とtask2が異なるゲームである実験3の方がtask1の学習結果に近くなっており、破滅的忘却が

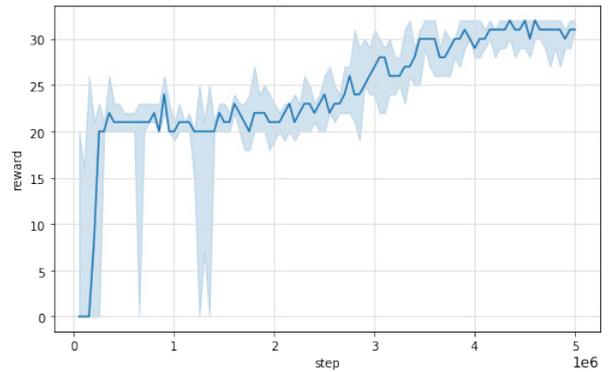


図6 実験 1,3 task1: Freeway 画像入力の学習結果

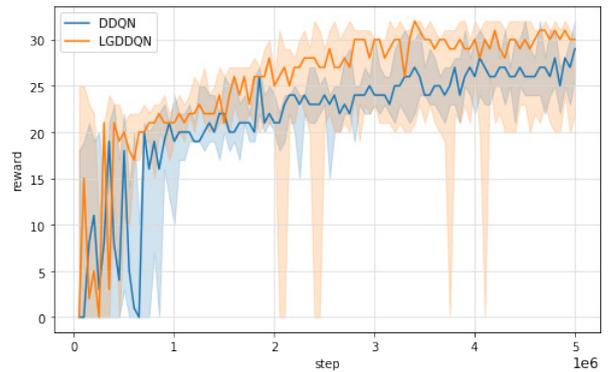


図7 実験 1 task2: Freeway 状態入力の学習結果

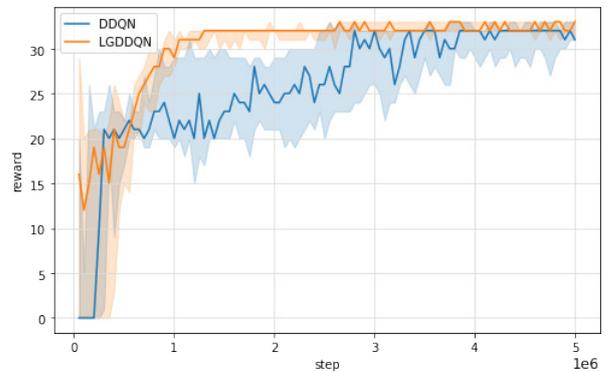


図8 実験 1 task3: Freeway 画像入力の学習結果

生じていることから、タスクが異なるほど破滅的忘却が生じやすいと考えられる。

提案手法のパラメータの選択は、実験1ではfc2-3でreuse、実験2ではfc2-3でreuse、fc4でadapt、実験3ではnewを多く選択しており、破滅的忘却を防ぐために、必要な分のパラメータを追加することができている。

また、各実験のtask2の学習結果の図7, 10, 12より、提案手法ではばらつきが大きい。これは、学習中にモデルの経路が変更されることや、選択が十分に収束しないまま経路を固定してしまったためと考えられる。

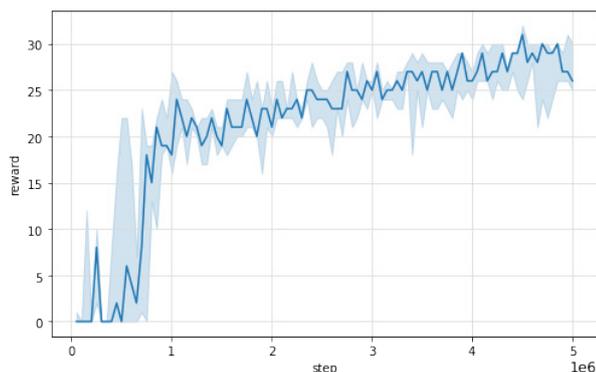


図 9 実験 2 task1: Freeway 状態入力の学習結果

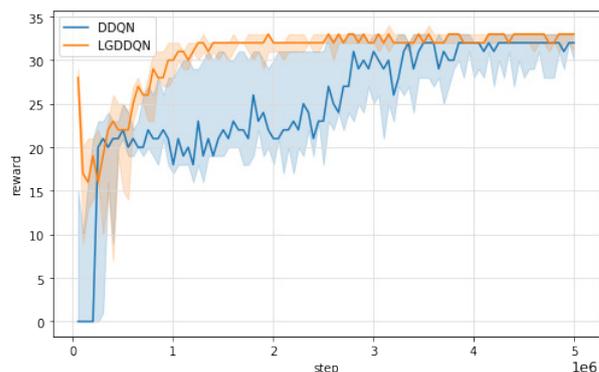


図 13 実験 3 task3: Freeway 画像入力の学習結果

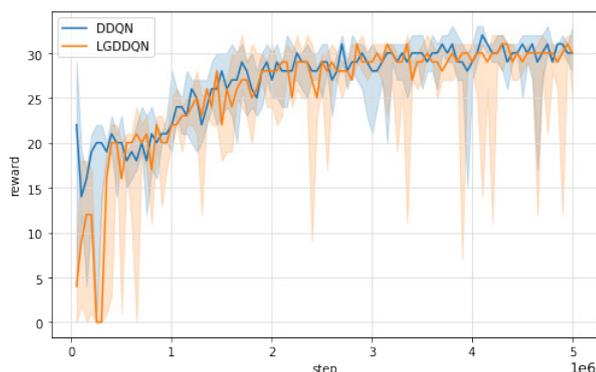


図 10 実験 2 task2: Freeway(行動変形) 状態入力の学習結果

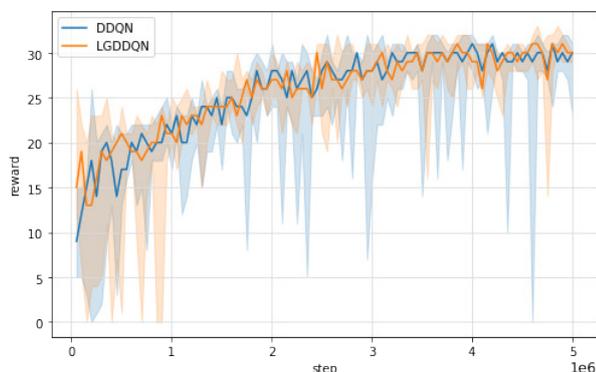


図 11 実験 2 task3: Freeway 状態入力の学習結果

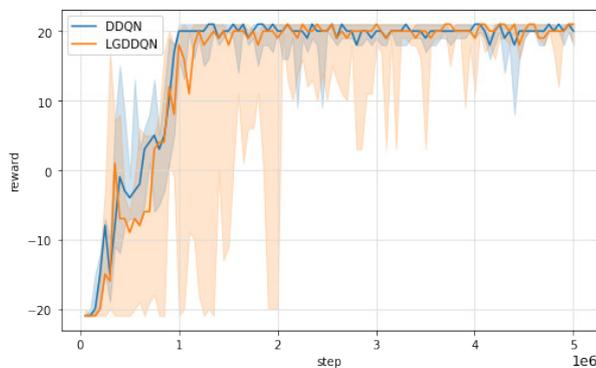


図 12 実験 3 task2: Pong 画像入力の学習結果

5. 終わりに

本稿では, OpenAI Gym のゲームを用いて, 画像入力と状態入力を混ぜた連続タスクで提案手法の有効性を検証した. その結果, 破滅的忘却を防ぐことができおり, 適切にパラメータを追加して学習が行えることを確認した. また, 一部のタスク間は学習済みのパラメータを利用して効率的な学習を行うことができた. 今後の課題としては, モデルの構造探索を改善して学習を安定化することが挙げられる. また, 今回のタスクは複数の入力形式を含んでいたが, 連続タスクとしては短いためより長い連続タスクでも有効か検証する必要がある.

参考文献

- [1] McCloskey, M., and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In Bower, G. H. (ed.), *The Psychology of Learning and Motivation*, Vol. 24, pp. 109-164. Academic Press, San Diego, CA, 1989. *Psychological review*, 97(2):285-308, 1990.
- [2] Pfülb, B., et al. Catastrophic forgetting: still a problem for DNNs. *Artificial Neural Networks and Machine Learning - ICANN*, 2018.
- [3] Ring, M. B., et al. Continual Learning in Reinforcement Environments. Ph.D. thesis, Univ. of Texas, 1994.
- [4] Kaplanis, C., et al. Continual Reinforcement Learning with Complex Synapses. *International Conference on Machine Learning (ICML)*, 2018.
- [5] Li, X., et al. Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting. *International Conference on Machine Learning (ICML)*, 2019.
- [6] Kirkpatrick, J., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13):3521-3526, 2017.
- [7] C. J. C. H. Watkins. Learning from delayed rewards. Ph.D. thesis, Univ. of Cambridge, UK, 1989.
- [8] Mnih, V., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529-533, 2015.
- [9] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., Learning representations by back-propagating errors. *Nature*, Vol. 323, pp. 533-536, October 1986.
- [10] Hasselt, H. V., Guez, A., and Silver, D. Deep Reinforce-

- ment Learning with Double Q-learning. *AAAI*, 2016.
- [11] Schaul, T., et al. Prioritized Experience Replay. *International Conference on Learning Representations (ICLR)*, 2016.
- [12] Rusu, Andrei, A., et al. Progressive neural networks. *arXiv:1606.04671*, 2016.
- [13] Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. *International Conference on Machine Learning (ICML)*, 2017.
- [14] Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning (ICML)*, 2017.
- [15] Nichol, A., Achiam, A., and Schulman, J. On first-order meta-learning algorithms. *arXiv:1803.02999*, 2018.
- [16] Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable Architecture Search. *International Conference on Learning Representations (ICLR)*, 2019.

付 録

A.1 モデルの構成

conv1 は 8×8 のカーネル, 32 のチャンネル, ストライドは 4 の畳み込み層, conv2 は 4×4 のカーネル, 64 のチャンネル, ストライドは 2 の畳み込み層, conv3 は 3×3 のカーネル, 64 のチャンネル, ストライドは 1 の畳み込み層, fc1 は画像入力では (3136, 512), 状態入力では (128, 512) の全結合層, fc2 は (512, 256), fc3 は (256, 128), fc4 は (128, output_size) の全結合層である. 活性化関数は全て ReLU を用いた.

reuse, new のパラメータは, それぞれの層でタスク毎に用いるパラメータと同じサイズである. adapt のパラメータは畳み込み層では 1×1 のカーネル, 同じ層の reuse と同じチャンネル, ストライドは 1 の畳み込み層である. 全結合層では, 例えば fc2 であれば (512, 512/16), (512/16, 256) のように層の入力を一度定数 (実験では 16) で割ったサイズ変換にする 2 つの全結合層をまとめたものである.

A.2 パラメータ設定

表 A.1 に実験に用いたパラメータ設定を示す.

設定	値
アルゴリズム	DDQN
ステップ数	5M
入力画像サイズ	84×84
フレームスタック	4
アップデート当たりの ステップ数	4
オプティマイザ	RMSprop
学習率	$2.5e-4$
オプティマイザの平滑化定数	0.95
オプティマイザの ϵ	$1e-2$
損失関数	Huber loss
パラメータ増加抑制係数 β	$1e-5$
L2 ペナルティ係数 λ	0
割引率 γ	0.99
構造探索停止のステップ数	$1e6$
リプレイメモリーのサイズ	$1e6$
マルチステップ学習のステップ数	5
行動選択アルゴリズム	EpsilonGreedy(start_eps=1, end_eps=0.1, decay_steps= $1e6$)
学習を開始するステップ数	$5e4$