修　　士　　論　　文

継続的強化学習における複数の入出力サイズを扱うモデルの学習

指導教員　　伊庭 斉志 教授

東京大学大学院情報理工学系研究科
電子情報学専攻

氏 名　48-206422 川島 丸生

提 出 日　　令和 4 年 1 月 27 日

概 要

人間のように他のタスクの経験を生かして新たなタスクを効率的に解決することを目指した手法として、継続的学習などの複数のタスクを扱う手法がある。複数のタスクを扱う手法は、様々な分野のタスクに対して、タスクを個別に学習するよりも効率的な学習が出来ているが、特に学習が難しい強化学習タスクに対しては、手法によって評価タスクが異なり、似たタスクのみを対象としている問題がある。より様々なタスクを含むベンチマークが考案されているが、センサなどの単一のモダリティによる入力形式のタスクに制限されている。本稿では、より様々なタスクを扱えるようにするために、強化学習タスクで多く用いられる入力形式である画像と状態の 2 つの入力を扱える、それぞれの入力を扱うモデルの中間層を共有したモデルを提案し、既存の複数のタスクを扱う手法である Learn to Grow や SoftModule を拡張して適用する。OpenAI Gym の Atari 環境のゲームを用いた画像入力と状態入力のタスクを含む連続タスクで、提案手法が過去のタスクの知識を失わずに活用して効率的に学習を進められることを確認した。

目 次

第 1 章 序論 **1**
1.1 背景 1
1.2 本研究の目的と貢献 2
1.3 本稿の構成 2

第 2 章 背景 **4**
2.1 強化学習 4
2.1.1 マルコフ決定過程 4
2.1.2 価値ベースの手法 7
2.1.3 Q 学習 7
2.1.4 Deep Q-Network (DQN) 9
2.1.5 方策ベースの手法 12
2.1.6 Actor-Critic 16
2.1.7 最大エントロピー強化学習 18
2.1.8 Soft Actor Critic（SAC） 20
2.2 複数のタスクを扱う手法 23
2.2.1 破滅的忘却 23
2.2.2 マルチタスク学習 23
2.2.3 メタ学習 24
2.2.4 継続的学習 26
2.2.5 マルチモーダル学習 27
2.2.6 強化学習タスク 28

第 3 章 関連手法 **30**
3.1 Learn to Grow 30
3.2 SoftModule 32

第 4 章 提案手法 **37**
4.1 基本モデル 37
4.2 Learn to Grow ベースの変更点 39
4.3 SoftModule ベースの変更点 40

第 5 章　評価実験 **41**
5.1　実験環境 41
5.1.1　Freeway 41
5.1.2　Pong 42
5.2　実験設定 42
5.2.1　モデルの構成 43
5.2.2　パラメータ設定 43
5.3　実験結果 48
5.3.1　実験 1: Freeway 画像-状態-画像 48
5.3.2　実験 2: Freeway 状態-状態 (行動変形)-状態 48
5.3.3　実験 3: Freeway 画像-Pong 画像-Freeway 画像 49

第 6 章　考察 **53**

第 7 章　結論 **55**
7.1　まとめ 55
7.2　今後の課題 55

図 目 次

2.1 各学習手法でのモデルのパラメータ変化の様子（[11] の Figure 1 を参考にした） . . 26
2.2 Space Invaders のプレイ画面 28
2.3 Boxing のプレイ画面 28
2.4 Ant の画面 28
2.5 HalfCheetah の画面 28
2.6 reach の画面 29
2.7 button-press のプレイ画面 29

3.1 Learn to Grow のモデル 30
3.2 Learn to Grow のある一層の変化の様子 31
3.3 SoftModule のモデル 34

4.1 提案手法の基本モデル（実線は画像入力の時の経路の例） 38
4.2 提案手法（LG ベース）の fc1-4 での各モジュールの生成方法 39

5.1 Freeway のプレイ画面 41
5.2 Pong のプレイ画面 41
5.3 学習中の提案手法（LG ベース）
のベースネットワーク 45
5.4 学習中の提案手法（LG ベース）
のルーティングネットワーク 45
5.5 提案手法（SM ベース）
のベースネットワーク 46
5.6 提案手法（SM ベース）
のルーティングネットワーク 46
5.7 実験 1 task1: Freeway 学習結果 50
5.8 実験 1 task2: Freeway-ram 学習結果 50
5.9 実験 1 task3: Freeway 学習結果 50
5.10 実験 2 task1: Freeway-ram 学習結果 51
5.11 実験 2 task2: Freeway-ram-shift 学習結果 51
5.12 実験 2 task3: Freeway-ram 学習結果 51

5.13 実験 3 task1: Freeway 学習結果 52
5.14 実験 3 task2: Pong 画像入力の学習結果 52
5.15 実験 3 task3: Freeway 画像入力の学習結果 52

表 目 次

5.1 各実験の task1 から task3（task3 は task1 と同一） 42
5.2 各層、モジュールの設定 44
5.3 実験に用いたパラメータ設定 47

第1章　序論

1.1　背景

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、効率的に新しいタスクを学習していく手法として、継続的学習（continual learning）[3] がある。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 [4] と、継続的学習を組み合わせた継続的強化学習（continual reinforcement learning） [5] 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。

継続的学習以外にも、複数のタスクを扱う手法としてマルチタスク学習 [6] やメタ学習がある。マルチタスク学習は、複数のタスクを同時に学習する手法で、タスク間の関係から共有可能な知識を獲得することで、個別にタスクを学習するよりも効率よく学習することが出来る。メタ学習は、与えられた複数のタスクを学習することで、新たなタスクに対して素早く学習できるようにする手法である。いずれの手法も、目標やタスク設定が異なるが、破滅的忘却に対処する必要のある複数のタスクを扱う手法であり、手法によっては他の手法として用いることが出来る。

継続的学習を含む複数のタスクを扱う手法の多くは、教師あり学習のタスクを扱うことが多く、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目標速度を変えて様々なタスクとして使用する [7]、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 [8] などの各手法独自のタスクで評価しており、限定的なタスク集合に対しての手法が多い。この問題に対処するために、ロボットアームのより様々なタスクからなるベンチマーク [9] などが提案されている。

しかし、既存の複数のタスクを扱う手法もベンチマークも、タスクからの入力は、あるロボットアームの決められたセンサの情報といった決められたモダリティによる入力情報、モーダルに制限されている。人が視覚や聴覚、嗅覚などの様々なモダリティを状況に応じて使い分けながら物事を認識するように、強化学習タスクも、例えばロボットアームの環境であればセンサによるアームの角度や速度だけでなく、カメラによるアームなどの画像、自然言語による命令などの様々なモダリティ

による入力情報が存在する。複数のタスクを扱う手法が、より様々なタスクを扱うためにも、様々な状況のタスクを扱えるようにするというタスクの内容の拡張だけでなく、複数のモダリティのタスクを扱えるようにするというタスクの形式の拡張をする必要がある。

複数のモダリティによるデータやタスクを扱う手法をマルチモーダル学習と呼ぶ。複数のモダリティのタスクを扱う、複数のタスクを扱う手法とマルチモーダル学習を合わせた手法として、カメラによる画像などのあるモダリティで学習したタスクの知識を、マイクによる音声といった異なるモダリティのタスクに用いる手法 [10] があるが、異なるモダリティ間で知識を共有することは難しく、新しい挑戦的な分野であるため先行研究は少ない。

1.2 本研究の目的と貢献

本研究の目的は、既存の複数のタスクを扱う手法が似通ったタスクから構成される狭い範囲のタスク集合に対してのみ適用されているという課題を解消して、より様々なタスクを効率的に学習することである。この課題に対して、既存の研究では各手法の評価のためのより多様なタスクを含んだベンチマーク [9] などが考案されているが、いずれも入力形式を統一したタスクに限定されている。人が物を視覚だけでなく触覚や嗅覚などの複数の方法で感じることが出来るように、強化学習タスクの環境を観測する方法は画面やカメラ、センサなど複数の方法がある場合が多く、複数の方法で観測するマルチモーダルなタスクは、既存の複数のタスクを扱う手法では入力形式を制限しているため扱うことが出来ない。そこで本稿では、既存の複数のタスクを扱う手法がより様々なタスクを扱えるようにするために、強化学習で良く用いられる入力形式である画像と状態の 2 つの入力を扱える手法の実現を目的とした。

本稿で提案する手法は、入力部分を画像と状態の 2 つに分けて隠れ層を共有したモデルに、既存の複数のタスクを扱う手法を拡張して適用することである。既存の手法として、画像入力の教師有り学習で評価を行った継続的学習手法の Learn to Grow [11] と状態入力の強化学習で評価を行ったマルチタスク学習手法の SoftModule [12] を用いた。評価実験として、画像入力と状態入力といった入力形式の異なるタスクを含んだ複数のタスクで評価を行う既存手法は存在しないため、二つの入力形式が可能な OpenAI Gym [13] の Atari 環境 [14] を用いて作成した連続タスクを作成し、提案手法の有効性を確認した。

1.3 本稿の構成

本稿の構成を以下に述べる。

まず、第 2 章では、本研究に関する前提知識について述べる。2.1 節では強化学習に関する基礎知識について説明し、2.2 節では複数のタスクを扱う手法に関する手法やタスクについて説明する。

続いて第 3 章では、本研究で用いる複数のタスクを扱う手法について説明する。

第 4 章では、本稿における提案手法について述べる。4.1 節では提案手法のモデルについて説明し、4.2、4.3 節では適用する手法の変更点について述べる。

第 5 章では、評価実験について述べる。5.1 節では評価実験の内容について、5.2 節ではモデルなどの詳細設定について、5.3 節では実験結果について述べる。

第 6 章では考察を述べる。

最後に第 7 章では、以上の内容を踏まえて、本稿の結論と今後の課題について述べる。

第2章　背景

2.1　強化学習

強化学習 [4] 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にどの行動を選択するかを学習する。

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　マルコフ決定過程

強化学習における環境は、マルコフ決定過程 (Markov decision process; MDP) としてモデル化されることが多い。MDP は、次の 4 要素の組 $< S, A, T, R >$ で表されるモデルである。

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

遷移関数 $T(s, a, s')$ は、ある状態 $s \in \mathcal{S}$ で行動 $a \in \mathcal{A}$ を選択して状態 $s' \in S$ に遷移する確率 $T(s, a, s') = \mathrm{Prob}\{s_{t+1} = s' \mid s_t = s, a_t = a\}$ である。また、報酬関数 $R(s, a, s')$ は、状態 $s \in \mathcal{S}$ で行動 $a \in \mathcal{A}$ を選択して状態 $s' \in \mathcal{S}$ に遷移した時に環境からエージェントに与えられる報酬である。遷移関数の条件が時刻 t の状態 s と行動 a しかないことから分かるように、時刻 t 以前の状態や行動の履歴に時刻 $t + 1$ の状態は依存しないことが分かる。この性質をマルコフ性と呼び、強化学習が適用される多くの問題の環境はマルコフ性を満たすとしてモデル化できる。環境のマルコフ性により、エージェントの各時刻における行動 a 選択の規則は、時刻 t の状態 $s_t \in \mathcal{S}$ を条件とした行動 $a_t \in \mathcal{A}$ の条件付確率分布 $P(a_t \mid s_t)$ で表現することが出来る。この行動選択の規則を方策（policy）π と呼び、方策 π に従って時刻 t の状態 s_t で行動 a_t を選択する確率を出力する場合は $\pi(s_t, a_t)$、方策 π に従って時刻 t の状態 s_t から行動 a_t を出力する場合は $\pi(s_t)$ と表す。

時刻 t の状態 $s_t \in S$ から、エージェントが方策 $\pi(s_t, a_t)$ に基づいて行動 $a_t \in A$ を取ると、環境が遷移関数 $T(s_t, a_t, s_{t+1})$ に基づいて次の状態 $s_{t+1} \in S$ に遷移して、エージェントは環境から報酬 $r_{t+1} = R(s_t, a_t, s_{t+1})$ を受け取る。強化学習では、この状態 s・行動 a・報酬 r を 1 ステップ（step）として繰り返して行く。また、初期状態 s_0 からいくつかの時間ステップを経て、修了状態 s_{terminal} に到着するなどによって行動が終了するまでの一連の履歴 $(s_0, a_0, r_1, s_1, a_1, r_2, \cdots)$ を 1 エピソード（episode）と呼ぶ。

強化学習の目的は、受け取る報酬の総和（累積報酬 R_t）を最大化するような方策 π を学習することである。累積報酬 R_t は次のように表される。

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \\ &= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \end{aligned} \tag{2.1}$$

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。

強化学習では、累積報酬 R_t を最大化する方策 π を得るために、現在の状態 s_t や行動 a_t がどの程度良いか、つまりどの程度の累積報酬 R_t が得られるかを示す指標として価値関数（value function）がある。状態の価値関数を状態価値関数（状態価値）と呼び、ある方策 π のもとで状態 s の価値は、その後に受け取る報酬の総和の期待値なので、

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}\left[R_t \mid s_t = s\right] \\ &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s\right] \end{aligned} \tag{2.2}$$

と表される。同様に、行動の価値関数を行動価値関数（行動価値）と呼び、ある方策 π のもとで状態 s において行動 a を選択する価値は、その後に受け取る報酬の総和の期待値なので、

$$\begin{aligned} Q^{\pi}(s, a) &= \mathbb{E}_{\pi}\left[R_t \mid s_t = s, a_t = a\right] \\ &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a\right] \end{aligned} \tag{2.3}$$

と表される。

価値関数は、定義より再帰的な関係式に書き直すことができる。

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}\left[R_t \mid s_t = s\right] \\ &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \,\middle|\, s_t = s\right] \\ &= \mathbb{E}_{\pi}\left[r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \,\middle|\, s_t = s\right] \\ &= \sum_a \pi(s,a)\sum_{s'} T(s,a,s')\left\{R(s,a,s') + \gamma\mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{(t+1)+k+1} \,\middle|\, s_{t+1} = s'\right]\right\} \end{aligned} \tag{2.4}$$

ここで、$\mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{(t+1)+k+1} \mid s_{t+1} = s'\right]$ は行動価値関数の定義より $V^{\pi}(s')$ となるので

$$V^{\pi}(s) = \sum_a \pi(s,a)\sum_{s'} T(s,a,s')\left\{R(s,a,s') + \gamma V^{\pi}(s')\right\} \tag{2.5}$$

のような関係式が得られる。この関係式を、状態価値関数のベルマン方程式と呼ぶ。同様に、行動価値関数の定義より

$$Q^{\pi}(s,a) = \sum_{s'} T(s,a,s')\left\{R(s,a,s') + \gamma\sum_{a'}\pi(s',a')Q^{\pi}(s',a')\right\} \tag{2.6}$$

行動価値関数のベルマン方程式が得られる。

状態価値関数 $V^{\pi}(s)$ は、ある方策 π のもとである状態 s からその後に受け取る報酬の総和の期待値なので、全ての状態 s において $V^{\pi}(s) \geq V^{\pi'}(s)$ となるとき、方策 π は π' より優れている。MDP では、任意の方策よりも優れているか同等な方策が存在することが保証されており、この方策を最適方策 π^* と呼ぶ。最適方策 π^* のもとでの状態価値関数を最適状態価値関数と呼び、

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{2.7}$$

と表される。同様に、最適方策 π^* のもとでの行動価値関数を最適行動価値関数と呼び、

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{2.8}$$

と表される。この最適状態価値関数 V^* と最適行動価値関数 Q^* のベルマン方程式をベルマン最適方程式と呼ぶ。それぞれのベルマン最適方程式はベルマン方程式と同様に導出され、

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')\left\{R(s,a,s') + \gamma V^*(s')\right\} \tag{2.9}$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s')\left\{R(s,a,s') + \gamma\max_{a'} Q^*(s',a')\right\} \tag{2.10}$$

と表される。

2.1.2 価値ベースの手法

強化学習の目的である累積報酬を最大化するような方策を得る方法として、価値関数を学習して価値関数によって方策を求める手法を、価値ベースの手法と呼ぶ。

価値関数を学習する手法として、ある時間ステップとその後の時間ステップの価値関数の差を利用して学習を行う TD（Temporal Difference）学習 [15] がある。TD 学習では、ある状態である行動をとったときの価値を、報酬と次の行動価値を複数サンプリングすることで期待値に収束させる。状態価値のベルマン方程式（式 2.5）より、状態 s の価値は報酬 $R(s,a,s')$ と次の状態 s' の価値からなる $R(s,a,s')+\gamma V^{\pi}(s')$ の期待値なので、単純に平均を取ると、k 個目の $R(s,a,s')$ と $V(s')$ が得られたときの状態価値 $V(s)$ の更新式は、

$$V(s) \leftarrow \frac{(k-1)\cdot V(s)+1\cdot\{R(s,a,s')+\gamma V(s')\}}{k} \tag{2.11}$$

となる。変形すると

$$Q(s,a) \leftarrow V(s)+\frac{1}{k}\left(R(s,a,s')+\gamma V(s')-V(s)\right) \tag{2.12}$$

となり、右辺の第一項が現在の推定値で第二項が新しく観測されたデータと現在の推定値の差に小さい値を掛けた値である。新しくサンプルされた価値を TD 目標（TD target）、新しくサンプルされた価値と現在の推定価値の差分 $R(s,a,s')+\gamma V(s')-V(s)$ を TD 誤差 δQ（TD error）と呼ぶ。TD 学習では、TD 誤差の係数は学習率やステップサイズ・パラメータ α という小さな正の値が用いられるため、更新式は

$$V(s) \leftarrow V(s)+\alpha\left[R(s,a,s')+\gamma V(s')-V(s)\right] \tag{2.13}$$

のようになる。この更新式では、1 ステップ先の価値を用いて更新を行っているが、さらに先の時間ステップの価値を利用することも可能であり、そのようなアルゴリズムを TD(λ) と呼び、式 2.13 の更新式のように 1 ステップ先の価値を用いる場合は $\lambda=0$ として TD(0) と呼ぶ。TD(0) アルゴリズムの疑似コードをアルゴリズム 1 に示す。

TD 学習を用いて行動価値関数を学習する手法として、SARSA（state-action-reward-state-action）[16] がある。SARSA では、エージェントの経験から得られる $(s_t,a_t,r_{t+1},s_{t+1},a_{t+1})$ の組を用いて、行動価値関数のベルマン方程式（式 2.6）より

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t)+\alpha\left[r_{t+1}+\gamma Q(s_{t+1},a_{t+1})-Q(s_t,a_t)\right] \tag{2.14}$$

のような更新式で行動価値関数 $Q(s,a)$ を学習する。SARSA は、エージェントの方策により得られた経験を用いて学習を行うため、方策オン型（On-policy）の手法に分類される。

2.1.3 Q 学習

TD 学習を用いて最適行動価値関数を学習する、強化学習の代表的な手法として Q 学習 [17] がある。行動価値関数のベルマン最適方程式（式 2.10）より TD 目標は $R(s,a,s')+\gamma\max_{a'}Q(s',a')$ と

アルゴリズム 1 TD(0) による状態価値関数 $V^\pi(s)$ の推定

Input: learning rate α, discount rate γ, policy π

Output: V^π

Initialize $V^\pi(s)$ arbitarily
for each episode **do**
  Reset environment, and observe initial state s
  while s is not terminal **do**
    $a \leftarrow \pi(s)$
    Take action a, observe reward r and next state s'
    $V^\pi(s) \leftarrow V^\pi(s) + \alpha\left[r + \gamma V(s') - V(s)\right]$
    $s \leftarrow s'$
  end while
end for

なる。時間ステップ t での状態 s_t でエージェントが行動 a_t を選択して、新しい状態 s_{t+1} と報酬 r_{t+1} が得られたときの経験 $(s_t, a_t, r_{t+1}, s_{t+1})$ を用いて、Q 学習では次のように更新を行う。

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right] \tag{2.15}$$

Q 学習は、十分な経験が得られていて学習率が一定の条件を満たす場合に、最適行動価値関数 Q^* に収束することが保証されている。

ランダムに行動を選択する方策であっても理論上は学習することが出来るが、ゲームなどの環境を考えると、ランダムな行動をするエージェントではゲームの終盤の経験が得られにくく、非常に効率が悪くなる。そのため、学習中の状態価値を用いた

$$\pi(s) = \arg\max_a Q(s, a) \tag{2.16}$$

のような greedy な方策を用いることで、より終盤の経験が得られるようになる。このような手法を貪欲法（greedy 法）と呼ぶ。しかし、greedy 法では行動価値が高い行動を常に選択するため、局所解に陥ってしまい、必要な経験が得られない可能性があるため、

$$\pi(s) = \begin{cases} \arg\max_a Q(s, a) & (\text{確率 } 1-\varepsilon) \\ \text{random action} & (\text{確率} \varepsilon) \end{cases} \tag{2.17}$$

のような確率 ε でランダムな行動を選択し、$1-\varepsilon$ で行動価値が最大となるような greedy な選択をする方策がよく用いられる。このような手法を ε-greedy 法と呼ぶ。ε-greedy 法の ε は、学習が進むにつれて行動価値が最適な値に近づくため、学習中に ε の値を徐々に小さくして greedy な行動による経験を得やすくすることが多い。Q 学習終了後は、学習がうまくいけば最適行動価値関数が得られるので、greedy な方策が最適方策となる。

Q 学習のアルゴリズムをアルゴリズム 2 に示す。SARSA と異なり、TD 目標 $r + \gamma \max_{a'} Q(s', a')$ の次の時間ステップの行動 a' は、エージェントの方策 π による選択 a_{t+1} と異なっていてもよいため、Q 学習は方策オフ型（off-policy）の手法に分類される。

アルゴリズム 2 Q 学習

Initialize $Q^{\pi}(s,a)$ arbitarily
for each episode **do**
  Reset environment, and observe initial state s
  while s is not terminal **do**
    Choose a from s using policy from Q (e.g., ε-greedy)
    Take action a, observe reward r and next state s'
    $Q^(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$
  end while
end for

2.1.4 Deep Q-Network (DQN)

SARSA や Q 学習では、行動価値関数 $Q(s,a)$ を学習するために、全ての状態と行動の組 (s,a) に対する価値をルックアップテーブルのようにそれぞれ保持する必要があり、状態や行動の数が大きい環境では現実的なリソースでは学習が困難である。そこで、行動価値関数 $Q(s,a)$ の推定値を直接保持するのではなく、ニューラルネットワークなどで価値関数を近似することで、より状態や行動の数が大きい環境での学習が可能となる。

Deep Q-Network (DQN) [18] は、深層ニューラルネットワークを用いて行動価値関数 $Q(s,a)$ を近似して Q 学習を行う、深層強化学習の手法である。DQN は、画像認識の分野でよく利用される Convolutional Neural Network（CNN）[19] を利用することで、ゲームなどの画像をそのまま状態として入力することが可能で、Atari 2600 の複数のビデオゲームにおいて人間よりも高いスコアを達成した。

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 $Q(s,a;\theta)$ と表現すると、Q 学習の最小化損失関数は

$$L(\theta) = \frac{1}{2}\left(r + \gamma \max_{a'} Q(s',a';\theta) - Q(s,a;\theta)\right)^2 \tag{2.18}$$

と表せるが、Q 学習と異なり、行動価値関数を近似しているため、学習する行動価値関数が最適行動価値関数に収束することが保証されない。そのため、DQN ではいくつかの工夫がなされている。

Experience Replay：強化学習の学習データは、エージェントが行動した順になっているため、得られた経験 $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$ をすぐに学習に用いると学習データ間に強い相関関係が出来るため、学習に悪影響がでる。そのため DQN では、得られた経験をリプレイメモリ $\mathcal{D}$（replay memory）に蓄積し、学習時にリプレイメモリからランダムにサンプリングした経験を用いるミニバッチ学習を行う経験再生（experience replay）[20] という手法を用いている。経験再生は DQN において最も性能に寄与している工夫である [21]。

Fixed Target Q-Network：ニューラルネットワークで近似した行動価値関数 $Q(s,a;\theta)$ の学習で、同じニューラルネットワークで近似した行動価値関数が TD 目標 $r + \gamma \max_{a'} Q(s',a';\theta)$ に含まれ

ていると、TD 目標と共に行動価値関数が振動する可能性があり学習に悪影響がでる。そのため DQN では学習を安定させるために、TD 目標に含まれる行動価値関数である Target Q-Network のミニバッチの学習中は θ^- に固定して、ミニバッチ完了後に $\theta^- \leftarrow \theta$ で更新する Fixed Target Q-Network という手法を用いている。

Reward Clipping：報酬は学習対象であるゲームなどの環境によってスケールが異なるため、同じパラメータで学習を行うと報酬の大きさによって勾配が大きく変化してしまい、学習が不安定になる可能性がある。そのため DQN では、報酬を負なら -1 正なら 1 のように $(-1, 0, 1)$ にクリッピングして報酬のスケールを揃えることで、勾配を安定化させる Reward Clipping という手法を用いている。

Huber loss：損失関数に二乗誤差を用いると、外れ値に対して勾配が極端に大きくなるため、DQN では

$$L(a) = \begin{cases} \frac{1}{2}a^2 & (|a| \leq \delta) \\ \delta(|a| - \frac{1}{2}\delta) & (|a| > \delta) \end{cases} \tag{2.19}$$

で表されるような Huber loss を用いる。δ は正のパラメータで、1.0 が使われることが多い。Huber loss は、$|a| > \delta$ の範囲では勾配が一定であるため、外れ値に対して鈍感であり、極端な勾配更新を防ぐことが出来る。

これらの工夫から、DQN の損失関数は

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}\left[\left(R(s,a,s') + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right] \tag{2.20}$$

となる。この誤差関数の勾配を用いて誤差逆伝播 [22] を行うことで、ニューラルネットワークの学習を行う。DQN のアルゴリズムを、アルゴリズム 3 に示す。

2.1.4.1 Double Deep Q-Network (DDQN)

Q 学習や DQN の TD 目標の第二項 $\gamma \max_{a'} Q(s', a')$ は、$\gamma Q(s', \arg\max_{a'} Q(s', a'))$ のように表せることから分かるように、Q 学習や DQN の TD 目標では、行動 a' を選択する行動価値関数と行動価値を評価する行動価値関数が同じであるため、環境の遷移や報酬が確率的であると行動価値の推定にノイズが入り、推定される行動価値が過大評価される問題 [23] がある。この問題に対処するため、Hasselt らは Q 学習を改善した Double Q-Learning [24]、DQN を改善した Double Deep Q-Network (DDQN) [25] と呼ばれる手法を提案した。Dboule Q-Learning と 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、それぞれに入るノイズが異なるものになるようにして、行動の過大評価を抑制した。DQN では、Fixed Target Q-Network により、学習対象の online network のパラメータ θ と target network のパラメータ θ^- の 2 つのネットワークが存在するため、DDQN では、行動を選択するネットワークのパラメータを θ^- から θ に置き換えることで行動を選択するネットワークと評価するネットワークが異なるようにした。DDQN の誤差関数は、以下のよ

アルゴリズム 3 Deep Q-Network（DQN）[18]

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random network parameters θ
Initialize target action-value function $\hat{Q}$ with network parameters $\theta^- = \theta$
for each episode **do**
  Reset environment, and observe initial state s_0
  for each step t of episode, state s_t is not terminal **do**
    $a_t \leftarrow \begin{cases} \text{random } a & \text{with probability } \varepsilon \\ \arg\max_a Q(s_t, a; \theta) & \text{otherwise} \end{cases}$
    Take action a_t, observe reward r_{t+1} and next state s_{t+1}
    Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $s_j, a_j, r_{j+1}, s_{j+1}$ from $\mathcal{D}$
    $y_j \leftarrow \begin{cases} r_{j+1} & \text{if } s_{j+1} \text{ is terminal} \\ r_{j+1} + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform gradient decent step on $(y_j - Q(s_j, a_j; \theta))^2$ w.r.t θ
    Every C steps reset $\hat{Q} \leftarrow Q$, i.e. set $\theta^- \leftarrow \theta$
  end for
end for

うになる。

$$L(\theta) = \mathbb{E}_{(s,a,r,s') \sim D} \left[\left(R(s,a,s') + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-) - Q(s,a;\theta) \right)^2 \right] \quad (2.21)$$

DQN は、深層強化学習の可能性を示した手法として非常に有名であり、DDQN 以外にも様々な改善を施した派生手法が発表されている。探索や学習効率を改善した手法としては、経験再生では珍しい経験の学習効率が低い問題を、経験の TD 誤差の大きさに応じてサンプルされる確率を高くすることで改善した優先度付き経験再生（Prioritized Experience Replay）[26] や、ε-greedy 法を用いると ε の変化のさせ方が性能を左右するハイパーパラメータとなる問題を、ε-greedy 法の代わりにネットワークにノイズを入れることで探索を行い ε の調整する必要を無くした Noisy Network [27]、優先度付き経験再生を拡張して分散処理により複数のエージェントを用いて効率的に経験を集め、各エージェントに異なる ε を用いて多様な探索を可能にした Ape-X [28] がある。DQN からネットワークを変えた手法としては、行動価値関数を状態価値とアドバンテージの和 $Q(s_t, a_t) = V(s_t) + A(s_t, a_t)$ として学習を行う Duleling Network [29] や、Ape-X に LSTM[30] を組み合わせた Recurrent Replay Distributed DQN（R2D2）[31]、Q-Network の出力を確率分布にして期待値ではなく分布を学習するようにした C51 [32]、DDQN や Dueling Network、Noisy Network、優先度付き経験再生、C51、Multi-step learning [4] を組み合わせた Rainbow [33] がある。

2.1.5 方策ベースの手法

価値関数を学習して価値関数によって方策を求める価値ベースの手法に対して、方策を直接学習する手法を、方策ベースの手法と呼ぶ。

方策を直接学習するために、方策のパラメータを θ として、パラメータ θ によって表される方策 π_θ を考える。初期状態 s_0 から方策 π_θ に従って t ステップ行動した時に、状態 $s_t = s$ となる確率を $Pr(s_t = s \mid s_0, \pi_\theta)$、状態 s となる定常確率を $d^\pi(s) = d^{\pi_\theta}(s) = \lim_{t\to\infty} Pr(s_t = s \mid s_0, \pi_\theta)$ とすると、方策ベースの手法の目的関数 $J(\theta)$ は、

$$J(\theta) = \sum_s d^\pi(s) V^\pi(s) = \sum_s d^\pi(s) \sum_a \pi_\theta(s,a) Q^\pi(s,a) \tag{2.22}$$

と定義される。ここで $V^{\pi_\theta}, Q^{\pi_\theta}$ は V^π, Q^π を表し、方策ベースの手法では累積報酬で割り引かない $\gamma = 1$ とする。方策のパラメータ θ を最適化するために、$\nabla_\theta J(\theta)$ を求める必要がある。まず $\nabla_\theta V^\pi(s)$ を考えると、

$$\begin{aligned}
\nabla_\theta V^\pi(s) &= \nabla_\theta \left(\sum_a \pi_\theta(s,a) Q^\pi(s,a) \right) \\
&= \sum_a \left(Q^\pi(s,a) \nabla_\theta \pi_\theta(s,a) + \pi_\theta(s,a) \nabla_\theta Q^\pi(s,a) \right) \\
&= \sum_a \left(Q^\pi(s,a) \nabla_\theta \pi_\theta(s,a) + \pi_\theta(s,a) \nabla_\theta \sum_{s'} T(s,a,s') \left(R(s,a,s') + V^\pi(s') \right) \right) \\
&= \sum_a \left(Q^\pi(s,a) \nabla_\theta \pi_\theta(s,a) + \pi_\theta(s,a) \nabla_\theta \sum_{s'} T(s,a,s') V^\pi(s') \right) \\
&= \sum_a Q^\pi(s,a) \nabla_\theta \pi_\theta(s,a) + \sum_a \pi_\theta(s,a) \sum_{s'} T(s,a,s') \nabla_\theta V^\pi(s')
\end{aligned} \tag{2.23}$$

と変形できる。ここで、状態 s から状態 x へ方策 π_θ のもとで k ステップ後に到達する確率を $\rho^\pi(s \to x, k)$ とすると、$k = 0$ の時は

$$\rho^\pi(s \to x, 0) = \begin{cases} 1 & (x = s) \\ 0 & (\text{otherwise}) \end{cases} \tag{2.24}$$

となり、$k = 1$ の時は

$$\rho^\pi(s \to x, 1) = \sum_a T(s,a,x) \pi_\theta(s,a) \tag{2.25}$$

となる。方策 π_θ のもとで状態 s から $k+1$ ステップ後に状態 x に到達する場合は、状態 s から k ステップで状態 s' まで行き、状態 s' から 1 ステップで状態 x に到達すれば良いので、

$$\rho^\pi(s \to x, k+1) = \sum_{s'} \rho^\pi(s \to s', k) \rho^\pi(s' \to x, 1) \tag{2.26}$$

のような ρ^π の関係式が得られる。簡単のため、$\phi(s) = \sum_a Q^\pi(s,a)\nabla_\theta \pi_\theta(s,a)$ として、ρ^π を用いて $\nabla_\theta V^\pi(s)$ をさらに変形すると、

$$
\begin{aligned}
\nabla_\theta V^\pi(s) &= \phi(s) + \sum_a \pi_\theta(s,a) \sum_{s'} T(s,a,s') \nabla_\theta V^\pi(s') \\
&= \phi(s) + \sum_{s'} \nabla_\theta V^\pi(s') \sum_a T(s,a,s') \pi_\theta(s,a) \\
&= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \nabla_\theta V^\pi(s') \\
&= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \left[\phi(s') + \sum_{s''} \rho^\pi(s' \to s'', 1) \nabla_\theta V^\pi(s'') \right] \\
&= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \phi(s') + \sum_{s''} \rho^\pi(s \to s'', 2) \nabla_\theta V^\pi(s'') \\
&= \phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \phi(s') + \sum_{s''} \rho^\pi(s \to s'', 2) \phi(s'') \\
&\quad + \sum_{s'''} \rho^\pi(s \to s''', 3) \nabla_\theta V^\pi(s''') \\
&= \cdots \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \rho^\pi(s \to x, k) \phi(s)
\end{aligned}
\tag{2.27}
$$

となる。目的関数の状態 s は初期状態 s_0 としても問題ないので、$\nabla_\theta J(\theta)$ は式 2.27 より

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta V^\pi(s_0) \\
&= \sum_s \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k) \phi(s) \\
&= \sum_s \phi(s) \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k)
\end{aligned}
\tag{2.28}
$$

と変形できる。ここで、$\eta(s) = \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k)$ と置くと

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \sum_s \eta(s) \phi(s) \\
&= \left(\sum_s \eta(s) \right) \sum_s \frac{\eta(s)}{\sum_s \eta(s)} \phi(s)
\end{aligned}
\tag{2.29}
$$

となる。$\sum_s \eta(s)$ は定数であり、$d^\pi(s)$ は定常確率を表すことから

$$
d^\pi(s) = \frac{\eta(s)}{\sum_s \eta(s)} \tag{2.30}
$$

のような関係式が成り立つので、

$$\begin{aligned}\nabla_\theta J(\theta) &\propto \sum_s \frac{\eta(s)}{\sum_s \eta(s)}\phi(s) \\ &= \sum_s d^\pi(s) \sum_a Q^\pi(s,a)\nabla_\theta \pi_\theta(s,a) \\ &= \sum_s d^\pi(s) \sum_a Q^\pi(s,a)\pi_\theta(s,a)\frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} \\ &= \sum_s d^\pi(s) \sum_a \pi_\theta(s,a) Q^\pi(s,a)\nabla_\theta \log(\pi_\theta(s,a)) \\ &= \mathbb{E}_{s\sim d^\pi, a\sim\pi_\theta}\left[Q^\pi(s,a)\nabla_\theta \log(\pi_\theta(s,a))\right]\end{aligned} \tag{2.31}$$

となる。簡単のため $\mathbb{E}_\pi = \mathbb{E}_{s\sim d^\pi, a\sim\pi_\theta}$ とし、必要なのは $J(\theta)$ の勾配方向なので比例式ではなく等式の形にすると

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi\left[Q^\pi(s,a)\nabla_\theta \log(\pi_\theta(s,a))\right] \tag{2.32}$$

となる。この式を方策勾配定理（Policy Gradient Theorem）[4] と呼ぶ。

方策勾配定理は、方策ベースの手法の基礎であるが、そのまま用いて勾配を計算すると、分散が大きくなりやすいという問題がある。そのため、式 2.32 に baseline という $b(s)$ 値を加えて

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi\left[\left(Q^\pi(s,a) - b(s)\right)\nabla_\theta \log(\pi_\theta(s,a))\right] \tag{2.33}$$

のようにして、$b(s)$ を $Q^\pi(s,a) - b(s)$ が小さくなるように調整することで勾配の分散を抑える方法がよく用いられる。

$$\begin{aligned}\mathbb{E}_\pi\left[b(s)\nabla_\theta \log(\pi_\theta(s,a))\right] &= \sum_s d^\pi(s)\sum_a \pi_\theta(s,a) b(s)\nabla_\theta \log(\pi_\theta(s,a)) \\ &= \sum_s d^\pi(s) b(s)\sum_a \nabla_\theta \pi_\theta(s,a) \\ &= \sum_s d^\pi(s) b(s)\nabla_\theta \sum_a \pi_\theta(s,a) \\ &= \sum_s d^\pi(s) b(s)\nabla_\theta 1 \\ &= 0\end{aligned} \tag{2.34}$$

であるので、baseline を追加することによって勾配の期待値は変化しない。

baseline を追加した方策勾配定理を用いて、方策のパラメータを最適化する基本的なアルゴリズムを Vanilla Policy Gradient と呼ぶ。Vanilla Policy Gradient では、方策 π_θ によって状態と行動の系列である軌跡 $\tau = (s_0, a_0, s_1, a_1, \cdots)$ を集め、方策のパラメータ θ と baseline のパラメータ ϕ の更新を繰り返す。$Q^\pi(s_t, a_t) = \mathbb{E}_\pi\left[R_t \mid s_t, a_t\right]$ なので、式 2.33 より

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_\pi\left[\left(Q^\pi(s,a) - b(s)\right)\nabla_\theta \log(\pi_\theta(s,a))\right] \\ &= \mathbb{E}_\pi\left[\left(R_t - b(s)\right)\nabla_\theta \log(\pi_\theta(s,a))\right]\end{aligned} \tag{2.35}$$

となるので、軌跡 $\tau_t^i = (s_t^i, a_t^i, s_{t+1}^i, \cdots)$ から計算される累積報酬を $R_t^i = \sum_{t'=t}^{\infty} r_{t'}^i$ とすると、勾配の推定値を $\hat{g} = \sum_i \sum_t (R_t^i - b(s_t)) \nabla_\theta \log \pi_\theta(s_t, a_t)$ として $\theta \leftarrow \theta + \alpha \hat{g}$ で更新を行えばよい。baseline は、$||b(s_t) - G_t^i||^2$ を最小とするように更新を行う。Vanilla Policy Gradient のアルゴリズムを、アルゴリズム 4 に示す。

アルゴリズム 4 Vanilla Policy Gradient

Initialize policy parameter θ_0, baseline parameter ϕ_0
for $k = 0, 1, 2, \cdots$ **do**
  Collect a set of trajectories $\mathcal{D}_k = \{\tau^i\}$ by policy π_θ
  Compute $R_t^i = \sum_{t'=t}^{\infty} r_{t'}^i$ for each trajectories τ^i
  Estimate policy gradient as
  $\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_i \sum_t (R_t^i - b_{\phi_k}(s_t^i)) \nabla_\theta \log \pi_\theta(s_t^i, a_t^i)$
  Compute policy update, either using standard gradient ascent,
  $\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k$
  Fit the baseline by regression on mean-squared error:
  $\phi_{k+1} = \arg\min_{\phi_k} \frac{1}{|\mathcal{D}_k|} \sum_i \sum_t ||b_{\phi_k}(s_t) - G_t^i||^2$
  typically via some gradient descent algorithm
end for

方策勾配定理の $Q(s, a)$ の近似や baseline $b(s)$ の更新方法は他にも様々な手法が存在する。REINFORCE [34] では、baseline b を報酬の平均、勾配の推定値 $\hat{g}$ を累積報酬 R_t を報酬 r_t で近似して、方策のパラメータ θ を次のように更新する。

$$\begin{aligned} b &= \frac{1}{|\mathcal{D}|} \sum_i \sum_t r_t^i \\ \hat{g} &= \frac{1}{|\mathcal{D}|} \sum_i \sum_t (r_t^i - b) \nabla_\theta \log \pi_\theta(s_t^i, a_t^i) \\ \theta &\leftarrow \theta + \alpha \hat{g} \end{aligned} \tag{2.36}$$

方策勾配定理の方策は確率的方策であるが、決定的な方策に対しても同様に勾配を導くことが出来る [35]。パラメータ θ によって表される決定的な方策を $a = \mu_\theta(s)$ とすると、目的関数は式 2.22 より

$$\begin{aligned} J(\theta) &= \sum_s d^\mu(s) V^\mu(s) = \sum_s d^\mu(s) Q^\mu(s, \mu(s)) \\ &= \mathbb{E}_{s \sim d^\mu} \left[Q^{\mu_\theta}(s, \mu_\theta(s)) \right] \end{aligned} \tag{2.37}$$

となる。勾配は、方策勾配定理と同様にして

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim d^\mu} \left[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu_\theta}(s, a) \Big|_{a = \mu_\theta(s)} \right] \tag{2.38}$$

となり、これを決定的方策勾配定理と呼ぶ。

2.1.6 Actor-Critic

方策の更新は、式 2.33 より、方策の分散を小さくするために baseline $b(s)$ を学習させる部分と、方策を学習する部分の 2 つに分けられる。baseline として価値関数 $V^\pi(s)$ を用いると、式 2.33 より

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\left(Q^\pi(s,a) - V(s) \right) \nabla_\theta \log(\pi_\theta(s,a)) \right] \tag{2.39}$$

となる。行動価値関数と状態価値関数の差をアドバンテージ関数 $A(s,a)$ と呼ぶ。

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \tag{2.40}$$

アドバンテージを用いると、方策勾配定理は

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[A^\pi(s,a) \nabla_\theta \log(\pi_\theta(s,a)) \right] \tag{2.41}$$

と表せる。方策により行動を決定して方策を更新する部分を行動器（Actor）、Actor の行動を評価して価値関数を更新する部分を評価器（Critic）と呼び、Actor と Critic を組み合わせた手法を Actor-Critic と呼ぶ。

Critic で TD 学習を用いて行動価値 Q の学習をする、TD 学習の SARSA と Actor-Critic を組み合わせた手法を TD actor-critic と呼ぶ。TD 学習では行動価値を次のステップの行動価値を用いて表すので、方策勾配は TD 誤差 $\delta = R(s,a,s') + \gamma Q_\phi(s',a') - Q_\phi(s,a)$ を用いて

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_\pi \left[\left(R(s,a,s') + \gamma Q_\phi(s',a') - Q_\phi(s,a) \right) \nabla_\theta \log(\pi_\theta(s,a)) \right] \\ &= \mathbb{E}_\pi \left[\delta_t \nabla_\theta \log(\pi_\theta(s,a)) \right] \end{aligned} \tag{2.42}$$

と表せる。TD actor-critic のアルゴリズムをアルゴリズム 5 に示す。

Critic では Q 学習で行動価値 Q を学習し、Actor では決定的な方策を用いる手法を、Deterministic Actor-Critic [35] と呼ぶ。Deterministic Actor-Critic は、決定的方策勾配定理を用いる Deterministic Policy Gradient（DPG）という手法の一種である。式 2.38 の決定的方策勾配法を用いて、Deterministic Actor-Critic での TD 誤差と各パラメータの更新は次のようになる。

$$\begin{aligned} \delta &= r + \gamma Q(s', \mu_\theta(s'); \phi) - Q(s,a;\phi) \\ \phi &\leftarrow \phi + \alpha_\phi \delta \nabla_\phi Q(s,a;\phi) \\ \theta &\leftarrow \theta + \alpha_\theta \nabla_\theta \mu_\theta(s) \nabla_a Q(s,a;\phi) \mid_{a=\mu_\theta(s)} \end{aligned} \tag{2.43}$$

決定的な方策と行動価値 Q を深層ニューラルネットワークで近似して学習を行う、DPG と DQN を組み合わせた手法を Deep Deterministic Policy Gradient（DDPG）[36] と呼ぶ。DDPG では、DQN と同様に学習を安定させるための工夫がなされている。

DQN の Target Network では、定期的にパラメータを同期しているが、パラメータを何ステップごとに同期するかが学習の安定性に寄与するハイパーパラメータとなる問題がある。そのため、DDPG では Target Network が Main Network を

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^- \tag{2.44}$$

アルゴリズム 5 TD Actor-Critic

Input: policy learning rate α_θ, action-value function learning rate α_ϕ, discount rate γ

Initialize policy parameter θ, baseline parameter ϕ
for each episode **do**
  Reset environment, observe initial state s, and sample action $a \sim \pi_\theta(s, \cdot)$
  for $t = 1, 2, \cdots$ **do**
    Sample next state $s' \sim T(s, a, s')$ and reward $r_t \sim R(s, a, s')$
    Sample next action $a' \sim \pi_\theta(s', \cdot)$
    Compute TD error for action-value at time t:
    $\delta_t = r_t + \gamma Q_\phi(s', a') - Q_\phi(s, a)$
    Use TD error to update the parameters of policy:
    $\theta \leftarrow \theta + \alpha_\theta \delta_t \nabla_\theta \log \pi_\theta(s, a)$
    Use TD error to update the parameters of action-value function:
    $\phi \leftarrow \phi + \alpha_\phi \delta_t \nabla_\phi Q_\phi(s, a)$
    Update $a \leftarrow a'$ and $s \leftarrow s'$
  end for
end for

のように緩やかに追従する soft target と言われる手法を用いる。τ は更新率であり、DDQN の論文では 0.001 としている。soft target と比較して、DQN で用いられていた同期方法を hard target と呼ぶ。

また、DQN では、探索を ε-greedy 法によって行っていたが、DDPG では

$$\mu'(s) = \mu_\theta(s) + \mathcal{N} \tag{2.45}$$

のように方策関数の出力にノイズを加えることで探索を行う。

DDPG のアルゴリズムをアルゴリズム 6 に示す。DDPG は、決定的な方策により行動空間が広くても比較的少ないサンプルで学習が可能であり、方策更新に用いる $Q(s, \mu(s); \phi)$ は再評価が可能なので DDPG は off-policy の手法であり DQN と同様に Replay Buffer を用いることが出来る。

DQN に対しては DDQN が行動価値 Q の推定を target q-network で行うことで過大評価を抑えたように、DDPG に対しても過大評価を抑える手法として Twin Delayed DDPG（TD3）[37] がある。TD3 では、DDPG の学習を安定させるために 3 つの工夫がなされている。

Clipped Double Q-Learning：TD3 では、行動価値関数をもう 1 つ同時に学習して行動価値が小さい方を用いることで過大評価を抑える Clipped Double Q-Learning という手法を用いる。この手法を加えると TD 目標は

$$r + \gamma \min_{i=1,2} \left(\hat{Q}_i(s', \hat{\mu}_{\theta^-}(s'); \phi^-) \right) \tag{2.46}$$

となる。

Target Policy Smoothing：TD3 では、行動価値関数を滑らかにしてノイズに対して強くするために、TD 目標の行動に平均 0 のガウスノイズを加える Target Policy Smoothing という手法を用

アルゴリズム 6 Deep Deterministic Policy Gradient（DDPG）[36]

Initialize actor network $\mu_\theta(s)$ and critic $Q(s,a;\phi)$ with parameters θ and ϕ
Initialize target network $\hat{\mu}$ and $\hat{Q}$ with parameters θ^- and ϕ^-
Initialize replay buffer $\mathcal{D}$
for each episode **do**
  Reset environment, observe initial state s_0
  for each step t of episode, state s_t is not terminal **do**
    Select action $a_t = \mu_\theta(s_t) + \varepsilon$, where $\varepsilon \sim \mathcal{N}$
    Take action a_t, observe reward r_{t+1}, and next state s_{t+1}
    Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of N transitions $s_j, a_j, r_{j+1}, s_{j+1}$ from $\mathcal{D}$
    Compute targets $y_j \leftarrow \begin{cases} r_{j+1} & \text{if } s_{j+1} \text{ is terminal} \\ r_{j+1} + \gamma \hat{Q}(s_{j+1}, \hat{\mu}_{\theta^-}(s_{j+1}); \phi^-) & \text{otherwise} \end{cases}$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_j (y_j - Q(s_j, a_j; \phi))^2$
    Update the actor policy using the sampled policy gradient:
    $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_j \nabla_\theta \mu_\theta(s_j) \nabla_a Q(s_j, a \mid \phi) \mid_{a=\mu(s_j)}$
    Update the target network: $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$, $\phi^- \leftarrow \tau\phi + (1-\tau)\phi^-$
  end for
end for

いる。TD3 の TD 目標は、2 つの手法を加えて

$$r + \gamma \min_{i=1,2} \left(\hat{Q}_i(s', \hat{\mu}_{\theta^-}(s') + \mathcal{N}(0,\sigma); \phi^-) \right) \tag{2.47}$$

となる。

Delayed Policy Update：行動価値関数に対して方策の更新は緩やかな傾向があるため、TD3 では Actor の更新頻度を Critic よりも下げる Delayed Policy Update という手法を用いている。TD3 の論文では、行動価値関数を 2 回更新するごとに方策を 1 回更新している。

2.1.7 最大エントロピー強化学習

強化学習では、データを集めるための探索（exploration）と報酬を最大化するための活用（exploitation）があり、両者のバランスを取ることが重要である。Q 学習においては、探索を行うために ε-greedy 方策を用いており、他にも Noisy Network のようなネットワークにノイズを加えることで探索を行う手法などがあるが、何れも確率的ノイズを加えることによって探索を促進する手法である。一方で、目的関数に方策のエントロピー項を加えることにより、報酬最大化だけでなく探索も目的とする強化学習を最大エントロピー強化学習と呼ぶ。強化学習の目的関数は

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right] \tag{2.48}$$

であるが、最大エントロピー強化学習ではエントロピー項を加えて

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha \mathcal{H}(\pi(s_t, \cdot))\right] \tag{2.49}$$

となる。ここで、α は温度パラメータと呼ばれるエントロピー項のハイパーパラメータで、$H(\pi(\cdot \mid s_t))$ は

$$H(\pi(s, \cdot)) = \sum_{a} -\pi(s, a) \log \pi(s, a) = \mathbb{E}_{a \sim \pi}\left[-\log \pi(s, a)\right] \tag{2.50}$$

のように表される方策のエントロピーである。

最大エントロピー強化学習における行動価値関数をソフト行動価値関数（soft Q-function）と呼び、次の式で定義される。

$$Q^{\pi}_{\mathrm{soft}}(s_t, a_t) = r_{t+1} + \mathbb{E}_{\pi}\left[\sum_{k=1}^{\infty} \gamma^k (r_{t+k+1} + \alpha \mathcal{H}(\pi(s_{t+k}, \cdot)))\right] \tag{2.51}$$

状態価値関数も同様にソフト状態価値関数と呼び、次の式で定義される。

$$V^{\pi}_{\mathrm{soft}}(s_t) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q^{\pi}_{\mathrm{soft}}(s_t, a)\right) da \tag{2.52}$$

最大エントロピー強化学習の最適方策は次の式で与えられることが知られている [38]。

$$\pi^*(s_t, a_t) = \exp\left(\frac{1}{\alpha}(Q^*_{\mathrm{soft}}(s_t, a_t) - V^*_{\mathrm{soft}}(s_t))\right) \tag{2.53}$$

$-\frac{1}{\alpha} Q_{\mathrm{soft}}(s_t, a_t)$ をエネルギーとすると、方策はボルツマン分布 $\pi(s_t, a_t) \propto \exp(-\mathcal{E}(s_t, a_t))$、$\frac{1}{\alpha} V_{\mathrm{soft}}(s_t)$ は分配関数と見なせる。ソフト行動価値関数とソフト状態価値関数に対してもベルマン方程式を考えることが出来、ソフトベルマン方程式は

$$Q_{\mathrm{soft}}(s_t, a_t) = r_{t+1} + \gamma \mathbb{E}_{s_{t+1} \sim T(s_t, a_t, \cdot)}\left[V_{\mathrm{soft}}(s_{t+1})\right] \tag{2.54}$$

となる。

最大エントロピー強化学習においてソフト行動価値関数を学習することで最適方策を求める手法として Soft Q-Learning（SoftQ）[38] がある。Soft-Q 学習では、ソフト行動価値関数を

$$\left(\hat{Q}_{\mathrm{soft}}(s_t, a_t; \phi^-) - Q_{\mathrm{soft}}(s_t, a_t; \phi)\right)^2 \tag{2.55}$$

が小さくなる方向に更新して、方策を

$$\arg\min_{\pi_\theta} \mathrm{D}_{\mathrm{KL}}\left(\pi_\theta(s_t, \cdot) \,\middle\|\, \exp\left(\frac{1}{\alpha}(Q_{\mathrm{soft}}(s_t, \cdot; \phi) - V_{\mathrm{soft}}(s_t; \phi))\right)\right) \tag{2.56}$$

のようにソフト行動価値関数から得られる Softmax 方策（ボルツマン方策）との KL 距離を最小化するように更新して近似することを繰り返して学習を行う。

2.1.8 Soft Actor Critic（SAC）

Soft-Q 学習は実装が非常に煩雑であり、方策に非常に表現力が高い関数を用いているため学習が不安定になりやすいという問題がある。そこで、Soft-Q 学習に様々な工夫を加えて学習を安定化した手法が Soft Actor Critic（SAC）[39] である。

SAC のソフト行動価値関数の TD 誤差は、式 2.54 のソフトベルマン方程式と TD3 で用いられた Clipped Double Q-Learning を用いて、サンプルした軌跡を $(s_j, a_j, r_{j+1}, s_{j+1})$ とすると、

$$\delta_j = r_{j+1} + \gamma \left[-\alpha \log \pi(s_{j+1}, a') + \min_{i=1,2} \hat{Q}_i(s_{j+1}, a'; \phi^-) \right] - Q(s_j, a_j; \phi) \tag{2.57}$$

となる。ターゲットパラメータの更新は、DDPG で用いられた soft target を用いている。方策の更新は、式 2.56 の Soft-Q 学習の方策の目標は

$$\begin{aligned}
&\arg\min_{\pi_\theta} \mathrm{D}_{\mathrm{KL}} \left(\pi_\theta(s_t, \cdot) \,\middle\|\, \exp\left(\frac{1}{\alpha} (Q_{\mathrm{soft}}(s_t, \cdot; \phi) - V_{\mathrm{soft}}(s_t; \phi)) \right) \right) \\
&= \arg\min_{\pi_\theta} \int_a \pi_\theta(s, a) \log \left(\frac{\pi(s, a)}{\exp\left(\frac{1}{\alpha} (Q_{\mathrm{soft}}(s_t, a; \phi) - V_{\mathrm{soft}}(s_t; \phi)) \right)} \right) da \\
&= \arg\min_{\pi_\theta} \mathbb{E}_{a \sim \pi_\theta(s, \cdot)} \left[\log \left(\frac{\pi(s, a)}{\exp\left(\frac{1}{\alpha} (Q_{\mathrm{soft}}(s_t, a; \phi) - V_{\mathrm{soft}}(s_t; \phi)) \right)} \right) \right] \\
&= \arg\min_{\pi_\theta} \mathbb{E}_{a \sim \pi_\theta(s, \cdot)} \left[\log \pi(s, a) - \frac{1}{\alpha} (Q_{\mathrm{soft}}(s_t, a; \phi) - V_{\mathrm{soft}}(s_t; \phi)) \right] \\
&= \arg\min_{\pi_\theta} \mathbb{E}_{a \sim \pi_\theta(s, \cdot)} \left[\alpha \log \pi(s, a) - Q_{\mathrm{soft}}(s_t, a; \phi) + V_{\mathrm{soft}}(s_t; \phi) \right] \\
&= \arg\max_{\pi_\theta} \mathbb{E}_{a \sim \pi_\theta(s, \cdot)} \left[Q_{\mathrm{soft}}(s_t, a; \phi) - \alpha \log \pi(s_t, a) \right]
\end{aligned} \tag{2.58}$$

と変形できるので、Clipped Double Q-Learning と合わせて

$$J_\pi(\theta) = \mathbb{E}_{a \sim \pi_\theta(s, \cdot)} \left[\min_{i=1,2} Q_i(s_t, a; \phi) - \alpha \log \pi(s_t, a) \right] \tag{2.59}$$

を最大化するように更新を行う。SAC は連続行動空間の環境を対象とした手法なので、Soft-Q 学習の表現力の高い方策関数の代わりに、ガウス方策関数を用いている。ガウス方策は、状態 s を入力として行動の平均 μ と標準偏差 σ を出力するが、行動 a を正規分布 $a \sim N(\mu, \sigma^2)$ からサンプルして得ると、方策の勾配が計算できなくなるため、標準正規分布のノイズ $z \sim N(0, 1)$ を用いて $a = \mu + \sigma z$ のように行動を求める Reparameterization Trick と呼ばれる手法を用いて、SAC では方策の勾配を計算できるようにしている。また、ガウス方策の行動は $-\infty$ から ∞ まで取りうるが、行動の範囲が制限されている環境もあるため、行動を tanh に適用して -1 から 1 の範囲に制限する Squashed Gaussian Policy と呼ばれる手法を用いてから環境による制限された行動の範囲に変換を行っている。tanh を適用する前の方策を $\mu(s, u)$、した後の方策を $\pi(s, a)$ とすると、

$$\log \pi(s, a) = \log \mu(s, u) - \sum_{i=1}^{D} \log(1 - \tanh^2(u_i)) \tag{2.60}$$

となる事を利用して $\log \pi(s,a)$ を計算することが出来る。

最大エントロピー強化学習では、温度パラメータ α がハイパーパラメータであり、行動価値関数やエントロピーは学習の進み具合によってスケールが変化するため、エントロピー項の係数である温度パラメータ α は適切に変化させる必要がある。SAC では、

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} R(s_t, a_t, s_{t+1})\right] \text{ s.t. } E_{\pi}\left[-\log(\pi(s_t, a_t))\right] \geq \tilde{\mathcal{H}} \tag{2.61}$$

のようにエントロピーの下限値 $\tilde{\mathcal{H}}$ の制約付き累積報酬和の最大化問題として α を決定する。この双対問題は

$$\max_{\pi} \min_{\alpha_t \geq 0} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} R(s_t, a_t, s_{t+1})\right] - \alpha_t \left(\mathbb{E}_{\pi}\left[-\log(\pi(s_t, a_t))\right] - \tilde{\mathcal{H}}\right) \tag{2.62}$$

となり、最適な双対変数 α_t は

$$\alpha_t^* = \arg\min_{\alpha_t} \mathbb{E}_{\pi}\left[-\alpha_t \log(\pi^*(s_t, a_t)) - \alpha_t \tilde{\mathcal{H}}\right] \tag{2.63}$$

なので、

$$J(\alpha) = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi(s_t, \cdot)}\left[-\alpha \log \pi(s_t, a_t) - \alpha \tilde{\mathcal{H}}\right] \tag{2.64}$$

を目的関数として、最小化するように α を更新する。この手法を Automatic Entropy Adjustment と呼ぶ。この式は、方策のエントロピーが設定した最小値よりも大きく確率的な時は α を小さくして方策が決定的になるようにし、逆の時は方策を確率的になるように更新することを意味している。エントロピーの最小値は、論文では行動の次元数を $|\mathcal{A}|$ として $\tilde{\mathcal{H}} = -|\mathcal{A}|$ のように設定している。SAC のアルゴリズムをアルゴリズム 7 に示す。

2.1.8.1 SAC-Discrete

連続行動空間での手法である SAC を離散行動空間に適用した手法が SAC-Discrete [40] である。離散行動空間では行動の次元数 $|\mathcal{A}|$ が有限であるため、SAC-Discrete では SAC と比較して、ソフト行動価値関数の入出力形式を $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ から $\mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$、方策の入出力形式を $\mathcal{S} \to \mathbb{R}^{2|\mathcal{A}|}$ から $\mathcal{S} \to [0,1]^{|\mathcal{A}|}$ に変更する。これにより、SAC では $\pi_\theta(s_j, \cdot), \pi_\theta(s_{j+1}, \cdot)$ からサンプリングして計算していた TD 目標 y_j や方策や温度パラメータの勾配を直接計算出来るため、推定値の分散を緩和することが出来る。

アルゴリズム 7 Soft Actor Critic（SAC）[39]

Initialize policy parameters θ and Q-function parameters ϕ_1, ϕ_2
Set target parameters equal to main parameters $\phi_1^- \leftarrow \phi_1, \phi_2^- \leftarrow \phi_2$
Initialize replay buffer $\mathcal{D}$ and a global temperature coefficient α
for each iteration **do**
  for each environment step **do**
    Sample next action $a_t \sim \pi(s_t, \cdot)$
    Take action a_t, observe reward r_{t+1}, and next state s_{t+1}
    Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $\mathcal{D}$
  end for
  for each gradient step **do**
    Sample random minibatch of transitions $B = (s_j, a_j, r_{j+1}, s_{j+1})$ from $\mathcal{D}$
    Compute targets for Q-functions:

$$y_j \leftarrow \begin{cases} r_{j+1} & \text{if } s_{j+1} \text{ is terminal} \\ r_{j+1} + \gamma \left(\min_{i=1,2} Q_i(s_{j+1}, a'; \phi_i^-) - \alpha \log \pi_\theta(s_{j+1}, a') \right) & \text{otherwise} \end{cases}$$

    where $a' \sim \pi_\theta(s_{j+1}, \cdot)$
    Update Q-functions by one step of gradient descent using
    $\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s_j, a_j, r_{j+1}, s_{j+1}) \in B} \left(y_j - Q(s_j, a_j; \phi_i) \right)^2$ for $i = 1.2$
    Update policy by one step of gradient ascent using
    $\nabla_\theta \frac{1}{|B|} \sum_{s_j \in B, a \sim \pi_\theta(s_j, \cdot)} \left(\min_{i=1,2} Q_i(s_j, a; \phi_i) - \alpha \log \pi_\theta(s_j, a) \right)$
    Update temperature coefficient by one step of gradient descent using
    $\nabla_\alpha \frac{1}{|B|} \sum_{s_j \in B, a \sim \pi_\theta(s_j, \cdot)} \left(-\alpha \log \pi_\theta(s_j, a) - \alpha \tilde{\mathcal{H}} \right)$
    Update the target network: $\phi_i^- \leftarrow \tau \phi_i + (1 - \tau) \phi_i^-$ for $i = 1, 2$
  end for
end for

2.2 複数のタスクを扱う手法

教師有り学習や強化学習などの機械学習では、特定のベンチマークのスコアなどの単一の指標に対する最適化、つまり 1 つのタスクに対する学習が一般的である。ここでのタスクとは、データセット $\mathcal{D}$ から特徴ベクトル x とラベル y のペアが与えられて

$$\mathcal{L}(\theta, \mathcal{D}) = -\mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\log f_\theta(y \mid x)\right] \tag{2.65}$$

のような損失関数が設定されたときに、

$$\min_\theta \mathcal{L}(\theta, \mathcal{D}) \tag{2.66}$$

のように損失関数を最小化するパラメータ θ を学習するような教師有り学習では、入力分布 $p(x)$ と入力に対する正解ラベルの分布 $p(y \mid x)$ と損失関数 $\mathcal{L}$ のセットのことで、

$$\mathcal{T} \triangleq \{p(x), p(y \mid x), \mathcal{L}\} \tag{2.67}$$

のようにタスク $\mathcal{T}$ が定義される。強化学習の場合は、状態空間 $\mathcal{S}$、行動空間 $\mathcal{A}$、初期状態分布 $p(s_0)$、遷移関数 $T(s, a, s')$ と報酬関数 $R(s, a, s')$ のセットで

$$\mathcal{T} \triangleq \{\mathcal{S}, \mathcal{A}, p(s_0), T(s, a, s'), R(s, a, s')\} \tag{2.68}$$

のように定義される。タスクの定義より、教師有り学習でデータが増えたり、強化学習で環境のダイナミクスや報酬が変化したりすると、別のタスクとなる。変化したタスクに対して、そのまま学習を続けると性能が低下したり、モデルやアルゴリズムの制約から学習を続けることができなくなるが、現実的な問題ではこのようなタスクの変化は多く、タスクの変化に対応したり、複数のタスクから共通の構造を見つけることで、新しいタスクに対して初めから学習することなく効率的に学習を行う様々な手法が考案されている。

2.2.1 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で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性能が初めからネットワークを学習させたときと同等かそれ以下に低下してしまう問題である。破滅的忘却が生じると、複数のタスクを個別に学習する方が性能や効率が良くなってしまうため、複数のタスクを扱う手法では必ず対処する必要がある問題である。

2.2.2 マルチタスク学習

1 つのタスクではなく、複数のタスクをまとめて学習する手法がマルチタスク学習（Multi-Task Learning; MTL）[6] である。マルチタスク学習では、複数のタスクから共有可能な表現や構造を獲得することで、個別にタスクを学習する通常の学習であるシングルタスク学習よりも、タスクの関連性などの一定の仮定の元でサンプル効率が高いことが示されている [41, 42]。

マルチタスク学習と比較するために、通常の L 層のニューラルネットワーク $f_\theta: \mathbb{R}^d \mapsto \mathbb{R}^k$ を考える。l 番目の隠れ層を h_l、隠れ層である初めの $L-1$ 層のパラメータを $\theta^{<L}$、ネットワークの表現である最後の隠れ層の出力を $\phi^T_{\theta^{<L}}: \mathbb{R}^d \mapsto \mathbb{R}^{h_{L-1}}$ とする。簡単のためにネットワークのヘッドである出力層のバイアスを省略して、出力層のパラメータを $w \in \mathbb{R}^{h_{L-1} \times k}$ とすると、入力 $x \in \mathbb{R}^d$ に対してネットワークの出力は

$$f_\theta(x) = \phi^T_{\theta^{<L}}(x) w \tag{2.69}$$

となる。多くのマルチタスク学習のネットワークは、複数のヘッドを持ち、各ヘッドが各タスクに対応している。このネットワークでは共有される $L-1$ 層の隠れ層が複数のタスクの共有表現として扱われ、$L-1$ 層の隠れ層をタスク毎に完全に共有したり、タスク毎に別のネットワークを用いてパラメータに制約を付けるなど、様々な手法が存在する [43]。入力を $x \in \mathbb{R}^d$、タスクに対応するヘッドのインデックスを $i \in [N]$ とすると、L 層 N ヘッドのニューラルネットワーク $\hat{f}_{\hat{\theta}}: \mathbb{R}^d \times [N] \mapsto \mathbb{R}^k$ の出力は

$$\hat{f}_{\hat{\theta}}(x, i) = \phi^T_{\hat{\theta}^{<L}}(x) \hat{w}^i \tag{2.70}$$

となる。ここで $\phi^T_{\hat{\theta}^{<L}}(x)$ は最後の隠れ層の出力、$\hat{\theta}^{<L}$ は初めの $L-1$ 層のパラメータ、$\hat{w}^i$ は出力層の i 番目のヘッド、ネットワークのパラメータ $\hat{\theta}$ は隠れ層とマルチヘッド出力層のパラメータの和であり $\hat{\theta} = \{\hat{\theta}^{<L}\} \cup \{\hat{w}^i\}_{i \in [N]}$ となる。マルチタスク学習では、タスクの集合からサブセット $\mathcal{B} \sim \{\mathcal{T}_i\}$ を取り、サブセット内のタスク $\mathcal{T}_k \in \mathcal{B}$ からいくつかのサンプル $\mathcal{D}^b_k \sim \mathcal{D}_k$ を取る。サンプルされたのタスク $\mathcal{T}_k$ のデータを $\mathcal{D}^b_k = \{(X_k, Y_k)\}$、損失関数を $\mathcal{L}_k$ とすると、マルチタスク学習の目的は

$$\begin{aligned} \min_{\hat{\theta}} \hat{\mathcal{L}}_{\mathrm{MTL}}(\hat{\theta}) &= \sum_{\mathcal{T}_k \in \mathcal{B}} \mathcal{L}_k(\hat{\theta}, \mathcal{D}^b_k) \\ &= \sum_{\mathcal{T}_k \in \mathcal{B}} \mathcal{L}_k \left(\phi^T_{\hat{\theta}^{<L}}(X_k) \hat{w}^k, Y_k \right) \end{aligned} \tag{2.71}$$

のように表される。

2.2.3 メタ学習

与えられた複数のタスクを学習することで、新たなタスクを素早く学習出来るようにする手法がメタ学習（Meta-Learning; ML）である。メタ学習は、与えられた複数のタスクのみを想定して未知のタスクへの適応を想定していないマルチタスク学習と異なり、複数のタスク（メタ訓練タスク; meta-training tasks）と適応する新しいタスク（メタテストタスク; meta-test task）が同じタスク分布から得られているという仮定があるが、未知のタスクへの適応を目的とした手法である。複数のタスクから新しいタスクを素早く学習するための方法を学習するため、メタ学習は学習法の学習（Learning to learn）とも呼ばれる。

メタ学習は、新たなタスクに少数のサンプルで適応するためにパラメータの最適化を工夫する最適化ベース（Optimization-Based）、外部メモリを用いる [44] などモデルを工夫するモデルベース（Model-Based）、タスクの入力データ間の計量（metric）を学習するメトリックベース (Metric-Based)

の大きく 3 つの手法に分けることが出来る。特に最適化ベースのメタ学習は、既存の勾配法による最適化を用いたモデルに用いることが出来るものが多く、様々な領域に応用されている。

メタ学習では、メタ訓練タスクやメタテストタスクを含むタスク分布 $p(\mathcal{T})$ からサンプルした訓練タスク $\mathcal{T}_k$ に対してメタ学習を行うことで、テストタスクに対してわずかなデータで適応できるようなパラメータや構造を獲得する。訓練タスク $\mathcal{T}_k$ から得たデータ $\mathcal{D}_k^b \sim \mathcal{D}_k$ を、訓練データ $\mathcal{D}_k^{\mathrm{tr}}$ とテストデータ $\mathcal{D}_k^{\mathrm{ts}}$ に分けると、最適化ベースのメタ学習の目的は

$$\begin{aligned}\min_\theta \mathcal{L}_{\mathrm{ML}}(\theta) &= \sum_{\mathcal{T}_k \sim p(\mathcal{T})} \mathcal{L}_k(\theta, \mathcal{D}_k^{\mathrm{tr}}, \mathcal{D}_k^{\mathrm{ts}}) \\ &= \sum_{\mathcal{T}_k \sim p(\mathcal{T})} \mathcal{L}_k(\theta - \alpha \nabla_\theta \mathcal{L}_k(\theta, \mathcal{D}_k^{\mathrm{tr}}), \mathcal{D}_k^{\mathrm{ts}})\end{aligned} \tag{2.72}$$

のように表される。

2.2.3.1 Model-Agnostic Meta-Learning

Model-Agnostic Meta-Learning (MAML) [7] は、メタ学習の最適化ベースの代表的な手法ので、タスク集合の各タスクに対して素早く学習が行われるようなパラメータを得ることができる。MAML では、タスク分布 $p(\mathcal{T})$ から取得したタスク $\mathcal{T}_i$ 毎に学習を行った場合に得られるパラメータの更新値 $\theta - \alpha \nabla_\theta \mathcal{L}_i(f_\theta)$ を計算し、そのパラメータでの損失のタスク集合からサンプリングされた全タスクに対する合計値が最小化するようにパラメータの更新を行う. パラメータの更新式は次のようになる。

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_i(f_{\theta - \alpha \nabla_\theta \mathcal{L}_i(f_\theta)}) \tag{2.73}$$

ここで、α, β は学習率である. MAML は、最も単純なマルチタスク学習と似ているが、マルチタスク学習はタスク集合からサンプリングしたタスクに対しての損失の期待値を最小化しており、MAML はタスクの評価時の損失の期待値を最小化している点で異なる。MAML は、タスクの評価時の損失を計算しているため、Hessian を求める必要があり計算コストが大きく、各タスクの評価時の少数のステップのうち、最後のステップでの勾配のみを用いることで、一次近似して計算コストを抑えた first-order MAML (FOMAML) [7] という手法がある。他にも、同様に二次勾配を無視して一次近似をするが、各タスクの評価時の勾配を全て合成するため、FOMAML と異なりステップ数が大きくても問題が生じず、FOMAML と同等の計算コストである Reptile [45] といったなどの様々な派生手法が存在する。

2.2.4 継続的学習

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が継続的学習（Continual Learning）である。継続的学習は、人が様々な課題を以前に学習した知識を活用しながら解決していく様子を模した問題設定であるため、生涯学習（Lifelong Learning）とも呼ばれる。継続的学習は、未知のタスクを想定しており学習中に一度に一つのタスクからしかデータを取ることが出来ないという点でマルチタスク学習と異なり、過去のタスクのデータをそのまま保存してはならないという点でメタ学習とも異なるが、一部のマルチタスク学習やメタ学習の手法は継続的学習の手法として用いることもできる。過去のタスクのデータをそのまま保存してはならないという制約は、継続的学習が現実的な様々なタスクを学習し続けるために、メモリリソースを制限して、ニューラルネットワークによる表現などで過去のタスクの知識を保持することを目標として課せられている。

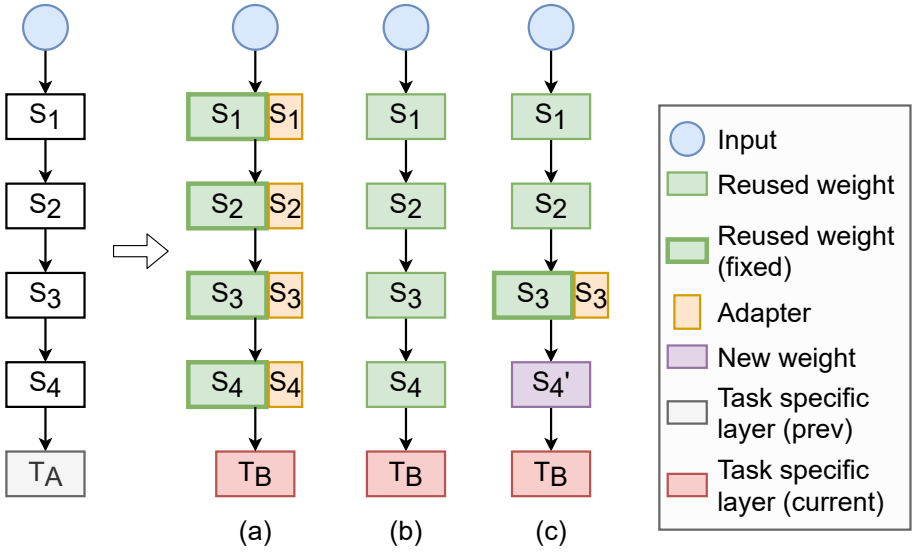

図 2.1: 各学習手法でのモデルのパラメータ変化の様子（[11] の Figure 1 を参考にした）

複数のタスクを逐次的に学習していき一度に一つのタスクにしかアクセス出来ないという継続的学習の設定は、複数のタスクのサブセットを同時にアクセスできるマルチタスク学習やメタ学習に比べて、破滅的忘却が問題となりやすい。破滅的忘却は、学習中のタスクによるパラメータの更新が過去のタスクの重要なパラメータを悪い方向に変化させてしまう場合に生じるため、勾配方向に制限をかけたり、過去のタスクを学習したパラメータを部分的に固定して新しいタスクのためのパラメータを増やすといった手法が存在する。タスク A の学習の後に新しいタスク B を学習する際に、モデルのパラメータの変化に注目すると、いくつかのマルチタスク学習、メタ学習や継続的学習の手法は、図 2.1 のように分類できる。

図 2.1 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 [46] 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 2.1 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) [8]、Zenke らの Synaptic Intelligence [47]、Finn らの Model-Agnostic Meta-Learning (MAML) や一次近似により MAML より計算量を減らした Nichol らの Reptile 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 2.1 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が、適切なパラメータを選択するという新たな問題を解決する必要がある。

2.2.5 マルチモーダル学習

マルチタスク学習、メタ学習や継続的学習は複数のタスクを扱う手法であるが、そのほとんどの手法が扱うタスクは、画像や言語、センサによる状態などのいずれか一つの単一のモダリティ（Modality）を入力したタスクの集合であり、画像を入力としたタスクや音声を入力としたタスクを含む複数のタスクや、画像と音声の両方を入力としたタスクといった複数のモダリティをもつタスクを扱うことはできない。モダリティとは、何かが生じたり何かを経験する方法のことで、私たちが物を見るだけでなく、触ったり匂いを嗅いだりと複数の方法で感じることが出来るように、身の回りのデータには複数のモダリティが存在しており、複数のモダリティが含まれているタスクやデータをマルチモーダル（Multimodal）と呼ぶ。人工知能が人間のように様々なタスクをこなすためには、マルチモーダルな情報を処理する必要があり、マルチモーダルなタスクを扱う手法がマルチモーダル学習（Multimodal Learning）である。

マルチモーダル学習のなかでも、強化学習タスクを扱う手法は、報酬設計をする代わりに言語入力で指示を出して迷路やビデオゲームのタスクを解く [48, 49] ような画像と言語を扱うものが多く、他にもビデオゲームの画像だけでなく BGM を入力として性能を向上させた [50] 画像と音楽を扱う手法なども存在する。

2.2.6 強化学習タスク

複数の強化学習タスクを扱う手法は多く存在するが、標準的なベンチマークが存在せず、各手法で異なった複数のタスクを扱っている。ここでは、複数の強化学習タスクを扱う手法で用いられている代表的な強化学習タスクを紹介する。

図 2.2: Space Invaders のプレイ画面

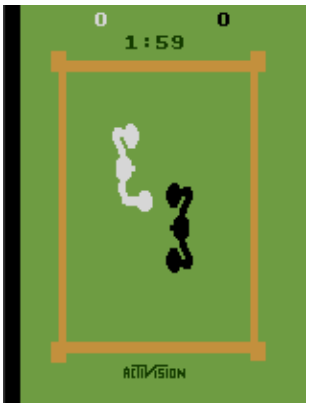

図 2.3: Boxing のプレイ画面

強化学習のタスクの多くは、OpenAI Gym [13] というツールキットに含まれているか、提供されるインターフェースに従っている。Arcade Learning Environment（ALE）[14] は、Atari 2600 という 1977 年に発売されたゲーム機をエミュレートして強化学習タスクとして提供するインターフェースであり、OpenAI Gym に含まれるタスクである。ALE のタスクは ALE 環境や、エミュレートしているゲーム機である Atari 環境と呼ばれる。ALE のゲームは、幅 160 ピクセル高さ 210 ピクセルの 7 ビットの 2 次元配列で表される 1 つの画面と、ジョイスティックコントローラーによる FIRE ボタンを押すか否か（2）とスティック上下左右斜めの 8 方向に倒すか否か（9）の組み合わせから最大計 18 個の行動から構成される。図 2.2, 2.3 に ALE 環境の一部のゲームのプレイ画面を示す。ALE 環境では、エージェントはプレイ画面のような画像以外にも、Atari 2600 本体のメモリである 128 バイトのメモリ情報を状態として受け取ることが出来るため、マルチモーダルなタスクを設定することが出来る。

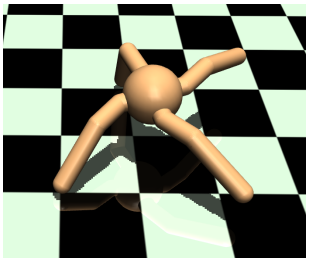

図 2.4: Ant の画面

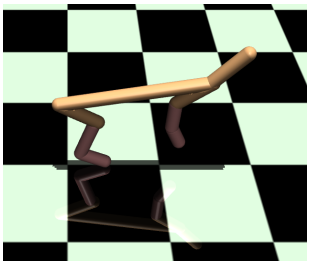

図 2.5: HalfCheetah の画面

OpenAI Gym には他にも、Multi-Joint dynamics with Contact（MuJoCo）[51] という物理演算エンジンを用いた 2/3 次元のロボットのタスクである mujoco 環境がある。mujoco 環境のタスクの一例として、図 2.4 に Ant という 3 次元の四足歩行ロボット環境の画面、図 2.5 に HalfCheetah という 2 次元のロボット環境の画面を示す。mujoco 環境では、図 2.4, 2.5 のような画像を出すこともできるが、関節の角度などのロボットのセンサに当たる情報を状態として、各関節にかける力を行動とする、連続行動空間のタスクとして設定されることが多い。複数のタスクを扱う手法である MAML では、Ant や HalfCheetah の目標速度を変えて複数のタスクとした設定で評価をしている。

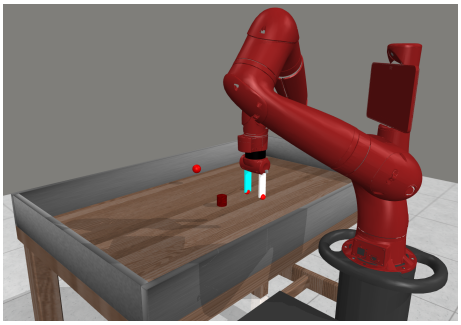

図 2.6: reach の画面

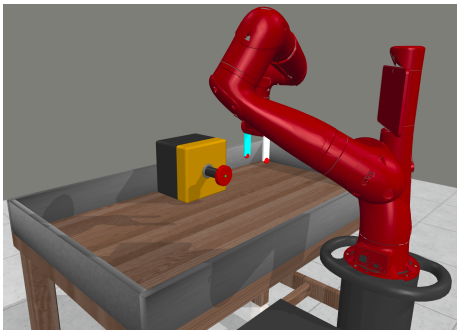

図 2.7: button-press のプレイ画面

より様々なタスクを含んだベンチマークとして、MuJoCo を用いたロボットアーム操作タスクセットである Meta-World [9] がある。Meta-World は、図 2.6, 2.7 のようにロボットアームを操作して物を把持して所定の場所へ運んだり、ボタンを押すといった様々な課題をこなすタスクで、50 種類の課題から構成されている。複数のタスクを扱う手法である SoftModule では、Meta-World のマルチタスク学習用の設定である MT50 で評価している。

第3章　関連手法

3.1　Learn to Grow

Learn to Grow [11] 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) [52] 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。

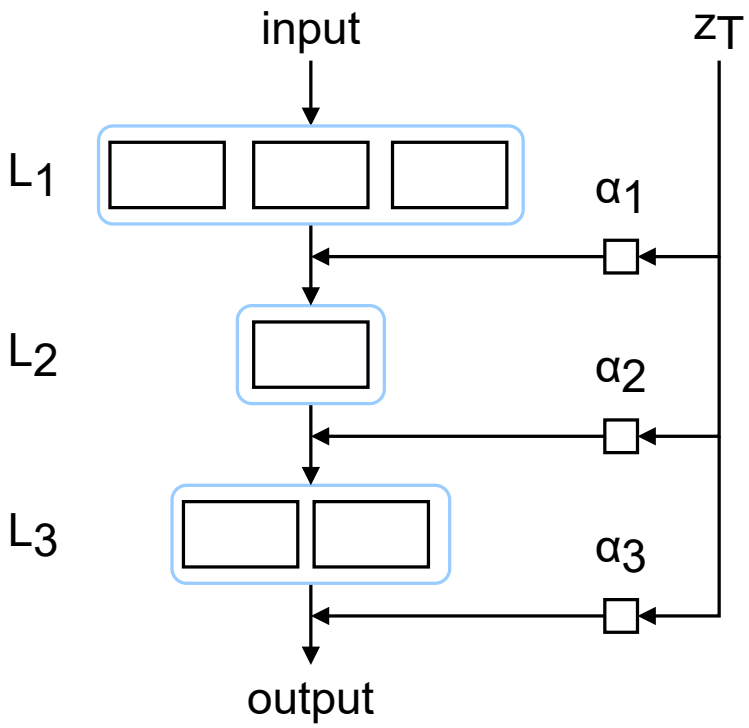

図 3.1: Learn to Grow のモデル

図 3.1 に 3 層で学習途中の Learn to Grow のモデルを示す。Learn to Grow のモデルは、畳み込み層や全結合層などをまとめてモジュールとして、いくつかのモジュールを持つ層と、各層のモジュールの選択の重みを決めるパラメータ α から構成される。Learn to Grow の学習は、図 3.2 のように

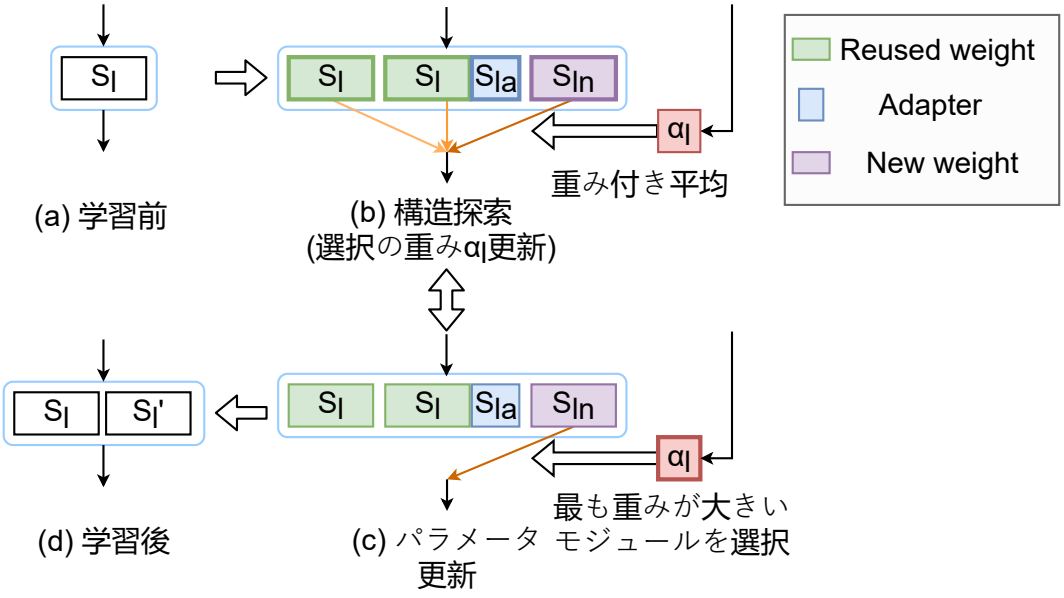

図 3.2: Learn to Grow のある一層の変化の様子

ネットワークの層毎のモジュールの選択を決定するための構造探索と、選択されたモジュールを更新するパラメータ更新の 2 つに分けられ、最終的に決定されたモジュール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 $\mathcal{T}_k$ を学習する際には層毎に既にあるモジュールをもとに、再利用（reuse）、小さな重みを追加（adaptation）、新しく生成（new）したモジュールを作成する。モデルの全てのパラメータを含むネットワーク S の第 l 層の過去のタスクで学習済みのモジュールの総数を $|S^l|$ と表すと、タスク $\mathcal{T}_k$ を学習中のネットワークの第 l 層のモジュールの総数は $C_l = 2|S^l|+1$ となる。

構造探索時、各層の出力は各モジュールの演算結果の重み付き和である。第 l 層の各モジュールの出力 g_c^l は入力を x_l とすると、reuse モジュール、adaptation モジュール、new モジュールで場合分けして

$$g_c^l(x_l) = \begin{cases} S_c^l(x_l) & (c \leq |S^l|) \\ S_c^l(x_l) + \gamma_{c-|S^l|}^l(x_l) & (|S^l| < c \leq 2|S^l|) \\ o^l(x_l) & (c = 2|S^l|+1) \end{cases} \tag{3.1}$$

と表される。ここで、$\gamma_{c-|S^l|}^l$ は adapt モジュールの追加された重みによる演算であり、o^l は new モジュールの追加された重みによる演算である。第 l 層の c 番目のモジュールの重みを α_c^l、モジュールの数を $C_l = 2|S^l|+1$ とすると、構造探索時の第 l 層の出力は

$$x_{l+1} = \sum_{c=1}^{C_l} \frac{\exp(\alpha_c^l)}{\sum_{c'=1}^{C_l} \exp(\alpha_{c'}^l)} g_c^l(x_l) \tag{3.2}$$

と表される。DARTS と同様に、層毎のモジュールの選択を全ての選択肢の出力のソフトマックスとするソフトな選択により、どのモジュールを選択するかという不連続的な探索空間から、選択肢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、第 l 層の出力は $c_l = \arg\max_c \alpha_c^l$ によりハードに選択されたモジュールの出力で

$$x_{l+1} = g_{c_l}^l(x_l) \tag{3.3}$$

と表される。

構造探索では各選択肢の重みである α_i^l のみ学習し、パラメータ更新時に選択されたモジュール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 $\mathcal{T}_k$ 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 $\mathcal{T}_k$ の学習に用いるパラメータ Θ_k 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 $s_k(\Theta_k)$ とすると、損失関数は

$$\mathcal{L}_k(s_k(\Theta_k)) = \frac{1}{n_k}\sum_{i=1}^{n_k} l_k(f(x_i^{(k)}; s_k(\Theta_k))、y_i^{(k)}) + \beta_k R_k^s(s_k) + \lambda_k R_k^p(\Theta_k) \tag{3.4}$$

と表される。ここで R_k^s は、構造探索において new モジュール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_k 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。第 l 層の c 番目の reuse モジュール以外のモジュールのサイズ（の対数）を z_c^l とすると $R_k^s(s_k) = \sum_{c>|S^l|,l} \alpha_c^l z_c^l$ と表される。R_k^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。Learn to Grow のアルゴリズムをアルゴリズム 8 に示す。

Learn to Grow の論文では、画像分類の異なる分野のタスクを集めた visual decathlon dataset（VDD）[53] や MNIST dataset [54] のような画像入力のタスクを扱っており、基本となるモデルは VDD に対しては ResNet [55] を用いている。モジュールは畳み込み層で、adapt モジュールの追加された重みである adapter には、reuse モジュールよりもパラメータ数が小さい演算として 1×1 の畳み込み層が用いられている。

3.2 SoftModule

SoftModule [12] は、マルチタスク学習手法の一つで、各層にいくつかのモジュールを保持して、タスクに応じてモジュール間の重みを決定するルーティングネットワークを用いた手法である。論文では、Meta-World [9] のマルチタスク学習用の設定である MT50 とその縮小版の MT10 で、マルチタスク学習用の SAC などと比較して性能を評価しており、MT50 では最も高い性能を出している。

アルゴリズム 8 Learn to Grow

Initialize model parameters Θ and module selection parameters α, replay buffer $\mathcal{D}$
for each task $\mathcal{T}_k$ in $[\mathcal{T}_1, \mathcal{T}_2, \cdots]$ **do**
  Initialize reuse, adapt, and new modules from learned modules in each layer
  Change to NAS mode
  for $iteration = 1, 2, \cdots$ **do**
    Sample data from $\mathcal{T}_k$ and store it in $\mathcal{D}$
    if $iteration\%(search\_step + parameter\_update\_step) < search\_step$ **then**
      Change to NAS mode
      Sample from $\mathcal{D}$ and compute loss (including parameter loss $R_t^s(s_t)$)
      Update $\alpha^{(k)}$
    else
      Change to Parameter Update mode
      Sample from $\mathcal{D}$ and compute loss
      Update $s_k(\Theta_k)$
    end if
  end for
end for

SoftModule は、マルチタスク学習の手法なので新しいタスクを学習する際にパラメータ数が変化しない手法である。各層にあらかじめ決めた数の畳み込み層全結合層などのモジュールを持ち、連続した 2 層のそれぞれのモジュール間をルーティングネットワークにより決められる重みで接続することで、各モジュールはそれぞれスキルや表現を学習し、ルーティングネットワークがタスクに応じて用いるモジュールをソフトに自動的に決定することが出来る。

$n = 3$ つのモジュールを持つ層が $L = 4$ 層の SoftModule のモデルを図 3.3 に示す。図 3.3 のように SoftModule は、通常の入力を処理するベースネットワークと、モジュールの選択を行うルーティングネットワークの 2 つのネットワークから構成される。通常の入力を x、各タスクを表す onehot ベクトルなどのタスクの埋め込みベクトルを $z_{\mathcal{T}}$ の 2 つを入力として、x は 2 層の多層パーセプトロン（Multilayer perceptron; MLP）から D 次元の表現 $f(x)$、$z_{\mathcal{T}}$ は 1 層の全結合層により同様に D 次元のタスクの埋め込み表現 $h(z_{\mathcal{T}})$ を得る。

ルーティングネットワークは、ベースネットワークのモジュールを持つ層と同じ深さで、ベースネットワークに n 個のモジュールを持つ層が L 層あるとすると、ルーティングネットワークの l 層目が出力するルーティングパラメータ p^l の次元は

$$p^l \in \begin{cases} \mathbb{R}^{n \times n} & (l \neq L) \\ \mathbb{R}^{1 \times n} & (l = L) \end{cases} \tag{3.5}$$

となる。ルーティングネットワークは、タスクと入力に応じてモジュールの選択を決めるために、入力の表現 $f(x)$ とタスクの埋め込み表現 $h(z_{\mathcal{T}})$ の要素積を 2 層の全結合層に通したものを入力とタスクの

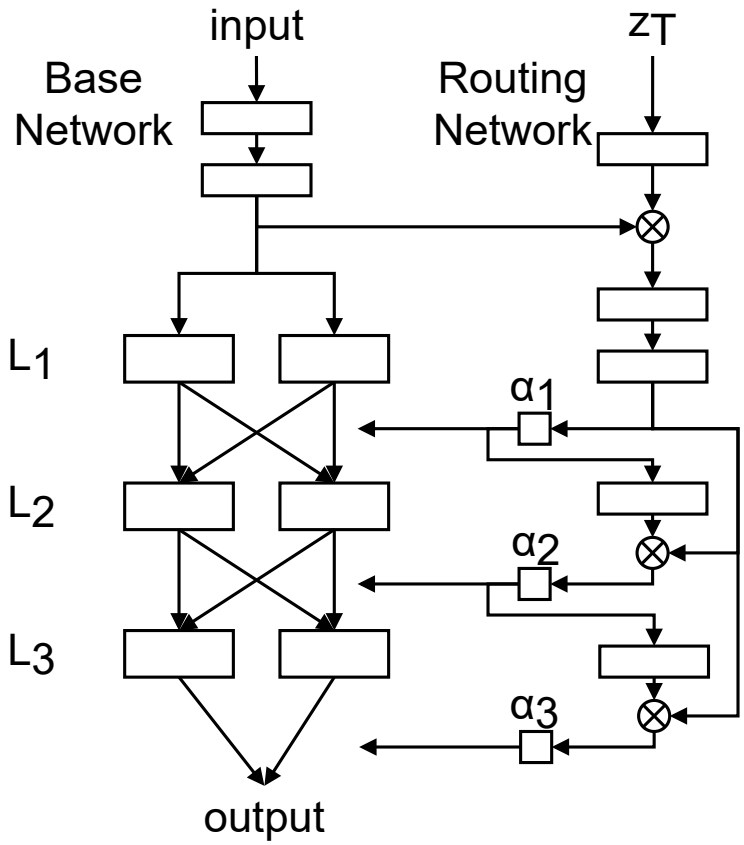

図 3.3: SoftModule のモデル

埋め込み表現 $g(f(x) \circ h(z_{\mathcal{T}})) \in \mathbb{R}^D$ として、前の層のルーティングパラメータを重み $\mathcal{W}_u^{l-1} \in \mathbb{R}^{D \times n^2}$ の全結合層に通して得られた選択の表現との要素積により得られた入力とタスクと選択の 3 つの情報を合わせた表現を全結合層により変換してルーティングパラメータ p^l が得られる。

$$p^l = \begin{cases} \mathcal{W}_d^l(\mathrm{ReLU}(g(f(x) \circ h(z_{\mathcal{T}})))) & (l = 1) \\ \mathcal{W}_d^l(\mathrm{ReLU}(\mathcal{W}_u^{l-1} p^{l-1} \circ g(f(x) \circ h(z_{\mathcal{T}})))) & (l \neq 1) \end{cases} \tag{3.6}$$

ここで $\circ$ は要素ごとの積、アダマール積を表す。$\mathcal{W}_d^l$ は、入力とタスクと選択の 3 つの情報を合わせた表現をルーティングパラメータに変換する全結合層の重みで、式 3.5 より

$$\mathcal{W}_d^l \in \begin{cases} \mathbb{R}^{n^2 \times D} & (l \neq L) \\ \mathbb{R}^{n \times D} & (l = L) \end{cases} \tag{3.7}$$

のような次元となる。ルーティングパラメータ p^l は、前の層の各モジュールの出力の加重平均を次

の層の各モジュールの入力とするための重みなので、ソフトマックス関数を用いて

$$\hat{p}_{i,j}^{l} = \frac{\exp(p_{i,j}^{l})}{\sum_{j=1}^{n} \exp(p_{i,j}^{l})} \tag{3.8}$$

のように正規化を行ってモジュールのソフトな選択を行う。

ベースネットワークは、モジュールを持つ層が L 層ありそれぞれ n 個のモジュールを持つ。第 l 層の第 j モジュールの入力を $u_j^l \in \mathbb{R}^D$ とすると、第 $l+1$ 層の第 i モジュールの入力は

$$u_i^{l+1} = \sum_{j=1}^{n} \hat{p}_{i,j}^{l}(\mathrm{ReLU}(\mathcal{W}_j^l u_j^l)) \tag{3.9}$$

と表される。ここで $\mathcal{W}_j^l \in \mathbb{R}^{D\times D}$ は第 l 層の第 j モジュールのパラメータを表す。$\hat{p}_{i,j}^{l}$ は正規化されているため、$\sum_{j=1}^{n} \hat{p}_{i,j}^{l} = 1$ となり、各モジュールの出力の加重平均となっている。最終層の出力を $o^L \in \mathbb{R}^o$ とすると、最終層の出力は

$$o^L = \sum_{j=1}^{n} \mathcal{W}_j^L u_j^L \tag{3.10}$$

となる。ここで $\mathcal{W}_j^L \in \mathbb{R}^{o\times D}$ は最終層 L の第 j モジュールのパラメータを表す。

SoftModule は Learn-to-Grow のモデルと層にいくつかのモジュールを持ちモジュールの選択も学習するという点で似ているが、ルーティングネットワークに当たる部分で状態や前の層のモジュールのルーティングパラメータと接続しており、タスク id だけでなく状態や前の層のモジュールのルーティングパラメータに応じたモジュールの選択が可能であり、モジュールの選択が出力の加重平均によるソフトな選択とどれか一つを選択するハードな選択が切り替わる Learn-to-Grow に対して常にソフトな選択であるという点で異なる。

複数のタスクを学習する際に、タスクによって収束の早さが異なるため、タスク間の学習のバランスを取る必要がある。SoftModule では、SAC を学習アルゴリズムに用いており、タスク毎の目的関数に重みをつけ、タスクの方策の信頼性が高いほど学習が進んでいるとして重みを小さく、信頼性が低いほど学習が進んでいないとして重みを大きくすることで学習のバランスを取っている。SAC では、式 2.64 エントロピーの最小値に対して $\log \pi(s_t, a_t)$ が大きくなるとエントロピーが小さくなるため温度パラメータ α を大きくして探索を促し、$\log \pi(s_t, a_t)$ が小さくなると α が小さくなるので、このタスクの目的関数の重みは、SAC の温度パラメータ α を用いて表すことができ、M 個の異なるタスクに対しての異なる温度パラメータを $\{\alpha_j\}_{j=1}^{M}$ とすると、タスク i の目的関数の重み w_i は

$$w_i = \frac{\exp(-\alpha_i)}{\sum_{j=1}^{M} \exp(-\alpha_j)} \tag{3.11}$$

のように表される。タスク $\mathcal{T}$ の Q 関数の目的関数を $J_{Q,\mathcal{T}}(\phi)$、方策の目的関数を $J_{\pi,\mathcal{T}}(\theta)$ とすると、式 2.71 のマルチタスク学習の目的から、SoftModule の Q 関数の目的関数は

$$J_Q(\phi) = \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}\left[w_{\mathcal{T}} J_{Q,\mathcal{T}}(\phi)\right] \tag{3.12}$$

となり、方策の目的関数は

$$J_\pi(\theta) = \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}\left[w_{\mathcal{T}} J_{\pi,\mathcal{T}}(\theta)\right] \tag{3.13}$$

となる。SoftModule のアルゴリズムをアルゴリズム 9 に示す。

アルゴリズム 9 SoftModule

Require: $p(\mathcal{T})$: distribution over tasks

Initialize policy parameters θ and Q-function parameters ϕ_1, ϕ_2

Set target parameters equal to main parameters $\phi_1^- \leftarrow \phi_1, \phi_2^- \leftarrow \phi_2$

Initialize replay buffer $\mathcal{D}$ and temperature coefficient $\{\alpha_i\}_{i=1}^M$ for each task

for each iteration **do**

  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$

  for all $\mathcal{T}_i$ **do**

    for each environment step **do**

      Sample next action $a_t \sim \pi(s_t, \cdot, z_{\mathcal{T}_i})$

      Take action a_t, observe reward r_{t+1}, and next state s_{t+1}

      Store transition $(s_t, a_t, r_{t+1}, s_{t+1}, z_{\mathcal{T}_i})$ in $\mathcal{D}$

    end for

  end for

  for each gradient step **do**

    Sample random minibatch of transitions $B = (s_j, a_j, r_{j+1}, s_{j+1}, z_{\mathcal{T}_i})$ from $\mathcal{D}$

    Compute task weight: $w_i = \frac{\exp(-\alpha_i)}{\sum_{j=1}^M \exp(-\alpha_j)}$

    Compute targets for Q-functions:

$$y_j \leftarrow \begin{cases} r_{j+1} & \text{if } s_{j+1} \text{ is terminal} \\ r_{j+1} + \gamma \left(\min_{i=1,2} Q_i(s_{j+1}, a', z_{\mathcal{T}_i}; \phi_i^-) - \alpha_i \log \pi_\theta(s_{j+1}, a', z_{\mathcal{T}_i}) \right) & \text{otherwise} \end{cases}$$

    where $a' \sim \pi_\theta(s_{j+1}, \cdot, z_{\mathcal{T}_i})$

    Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s_j, a_j, r_{j+1}, s_{j+1}, z_{\mathcal{T}_i}) \in B} w_i \left(y_j - Q(s_j, a_j, z_{\mathcal{T}_i}; \phi_i) \right)^2 \quad \text{for } i = 1.2$$

    Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{(s_j, z_{\mathcal{T}_i}) \in B, a \sim \pi_\theta(s_j, \cdot, z_{\mathcal{T}_i})} w_i \left(\min_{i=1,2} Q_i(s_j, a, z_{\mathcal{T}_i}; \phi_i) - \alpha_i \log \pi_\theta(s_j, a, z_{\mathcal{T}_i}) \right)$$

    Update temperature coefficient for each task by one step of gradient descent using

$$\nabla_{\alpha_i} \frac{1}{|B_i|} \sum_{(s_j, z_{\mathcal{T}_i}) \in B, a \sim \pi_\theta(s_j, \cdot, z_{\mathcal{T}_i})} \left(-\alpha_i \log \pi_\theta(s_j, a, z_{\mathcal{T}_i}) - \alpha_i \tilde{\mathcal{H}} \right)$$

    Update the target network: $\phi_i^- \leftarrow \tau \phi_i + (1 - \tau) \phi_i^- \quad \text{for } i = 1, 2$

  end for

end for

第4章　提案手法

2.2 節で紹介したように、複数のタスクを扱う手法は多く存在するが、ほとんどが画像やセンサによる状態などの単一のモダリティでの固定された入力形式を扱う手法であり、扱うタスクも目標速度を変化させるといった狭い範囲の複数のタスクである。ロボットアームによる、より様々なタスクを含んだベンチマーク [9] も提案されているが、入力はロボットアームのセンサによる状態入力のみであり、複数のモダリティを扱う手法も存在するが、画像入力に加えて言語入力を補助的に用いるといった補助的に入力を追加する手法が多い。人が視覚だけでなく触覚や聴覚などで物事の情報を得るように、現実的なタスクでも複数のモダリティから情報を得ることができる。複数のタスクを扱う手法の目的である、人間のように複数のタスクの経験から有用な経験を利用して新たなタスクに対処しつつ過去のタスクの経験を忘れないためには、学習効率などの改善だけでなく、様々なモダリティを扱えるようにする必要がある。

そこで本研究では、より様々なタスクを扱うことを目的として、強化学習で扱われることの多い 2 つのモダリティである、画像入力と状態入力のどちらも扱うことが出来る手法を提案する。

4.1　基本モデル

2.2 節の図 2.1 で分類されるように、複数のタスクを扱う手法でパラメータの量を変化させる手法はいくつかあるが、いずれも通常の 1 つのタスクを学習するときのモデルを基本として、第 i 層で処理した表現を第 $i+1$ 層に送るという流れを変えずに、各層毎にモジュールなどを追加するという手法である。通常の 1 つのタスクを学習する場合のモデルを、本稿では基本モデルと呼ぶ。複数のタスクを扱う手法は、基本モデルを元に各層のパラメータの量や制約を変化させて学習を行うため、ほとんどの手法では扱うタスクを 1 つにした場合や継続的学習での初めのタスクを学習する場合のモデル構造は基本モデルと同じとなる。

提案手法の基本モデルを図 4.1 に示す。画像入力では畳み込み層の Conv1-3 の 3 層を通して後に 1 次元に変換して全結合層の fc1 (img) を通して、状態入力では全結合層の fc1 (ram) を通して入力を変換してから、全結合層の fc2-3 の 2 層により表現を得る。得られた表現は、マルチタスク学習のマルチヘッドモデルと同様に、出力形式に対応したヘッドである全結合層の fc4 に通してモデルの出力が得られる。画像入力のモデルと状態入力のモデルの中間層を共有することで、異なるモダリティによる入力に対する共有表現を学習することが出来る。

Conv1-3 と fc1 (img) を画像入力に対するエンコーダ、fc1 (ram) を状態入力に対するエンコーダと見ると、扱うモダリティは異なるものの、各モダリティに対するエンコーダの出力を結合して共同表現を得るマルチモーダル学習の手法 [56, 57, 58, 59] と似ている。マルチモーダル学習の手法で

は、他のモダリティによる情報を補助的に用いるために、1 つのタスクで複数のモダリティによる入力が存在するため、各エンコーダの出力の和や結合を用いるが、本稿では基本的なタスクと同様に 1 つのタスクの入力は 1 つのモダリティからのみであるとして、過去のタスクの知識を活用するために、1 つのタスクに対して何方か一方のエンコーダの出力を用いるという点で異なる。本稿では、第 5 章で説明するように、各タスクの入力はそれぞれ 1 つのモダリティからのみであるため、入力形式に応じて切り替えているが、ロボットアームなどのタスクで、複数のモダリティによる入力があるタスクとないタスクを含めた複数タスクなども考えられ、各エンコーダの出力の和を用いるなど拡張することも出来る。

提案手法では、基本モデルの各層にモジュールを持ち、それぞれのモジュールの選択を行うルーティングネットワークを用いる。モジュールの生成やその選択は、2.2.4 節で紹介したように様々な方法があるが、本研究では 3 章で紹介した比較的性能と効率のよい Learn to Grow（LG）[11] と SoftModule（SM）[12] の二つの手法をそれぞれ用いる。

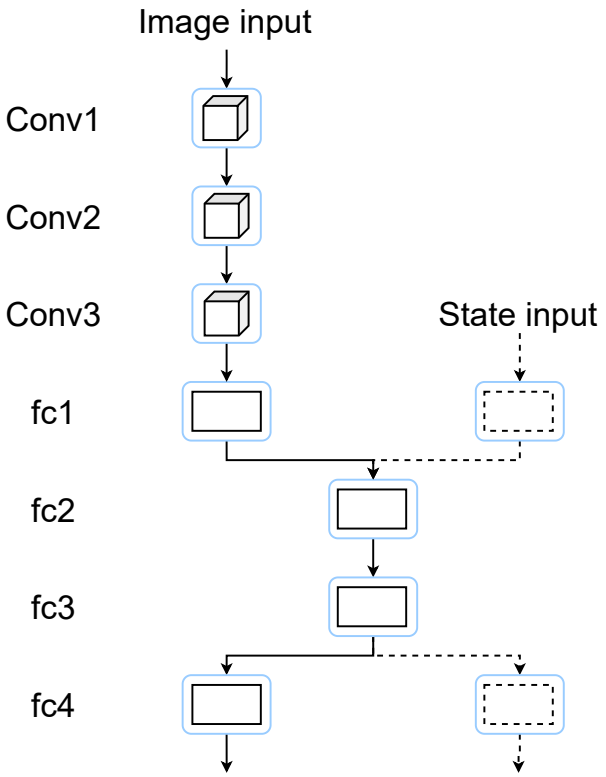

図 4.1: 提案手法の基本モデル（実線は画像入力の時の経路の例）

4.2 Learn to Grow ベースの変更点

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 モジュールの adapter はカーネルサイズ 1 の畳み込み層であり、全結合層には用いることができない。そのため、LG ベースの提案手法における全結合層の adapter 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、図 4.2 のような 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。

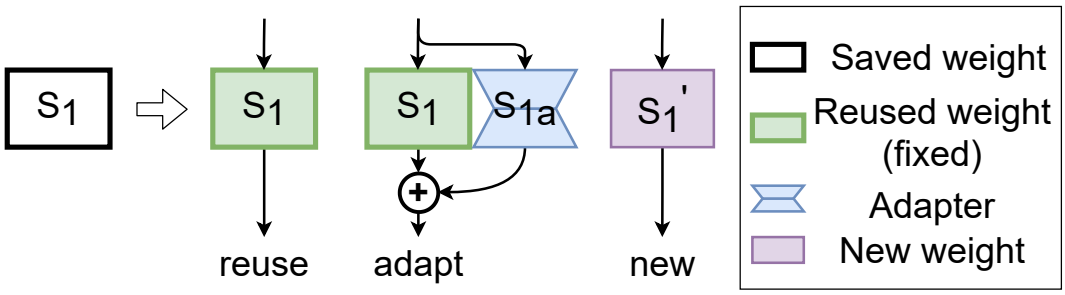

図 4.2: 提案手法（LG ベース）の fc1-4 での各モジュールの生成方法

学習中の LG では、各層の出力が学習中のタスクで扱うモジュールの出力のソフトマックスとなるソフトなモジュール選択を行う構造探索と、選択肢の重み α が最も大きいモジュールの出力となるハードなモジュール選択を行うパラメータ更新を繰り返し行う。LG は教師有り学習のタスクで評価されていたが、提案手法では強化学習のタスクで評価するため、選択されている経路のモジュール評価が行えるように、エージェントが環境に対して実際に行動を決定する時には、パラメータ更新モードとして、モデルの各層の出力を選択肢の重みが最も大きいモジュールの出力とした。

モデルやパラメータの更新は、強化学習の更新アルゴリズムにおけるネットワーク更新時に行う。教師有り学習と異なり、強化学習ではエージェントのモデルの出力が学習データに影響するため、ソフトな選択とハードな選択の切り替えのようにモデルを大きく変化させると、学習が非常に不安定になる可能性がある。そのため、1 つのタスクに対して一定ステップまでは構造探索とパラメータ更新を繰り返し、一定ステップ後はパラメータ更新のみとし、モジュール選択肢の重み α を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。LG ベースの提案手法におけるアルゴリズムのモデル更新部分をアルゴリズム 10 に示す、初期化などの処理は Learn to Grow のアルゴリズム 8 と同様である。

アルゴリズム 10 提案手法（LG ベース）におけるモデルの更新

Require: model parameters Θ, model selection parameters α, replay buffer $\mathcal{D}$, task $\mathcal{T}_k$

for $iteration = 1, 2, \cdots$ **do**
  Sample data from $\mathcal{T}_k$ and store it in $\mathcal{D}$
  if $iteration < max\_NAS\_step$ and $iteration\%(search\_step + parameter\_update\_step) < search\_step$ **then**
    Change to NAS mode
    Sample from $\mathcal{D}$ and compute loss (including parameter loss $R_t^s(s_t)$)
    Update $\alpha^{(k)}$
  else
    Change to Parameter Update mode
    Sample from $\mathcal{D}$ and compute loss
    Update $s_k(\Theta_k)$
  end if
end for

4.3 SoftModule ベースの変更点

SM では、入力 x を 2 層の全結合層に通して入力の表現 $f(x)$ としてベースネットワークとルーティングネットワークに送っていたが、この 2 層の全結合層は全てのタスクに対して同じものが使われるため、継続的学習のタスク設定では特に全く異なるタスクを学習した際に破滅的忘却が生じて、入力の表現が大きく変化してしまいルーティングネットワークにも影響を及ぼす可能性がある。そのため、SM ベースの提案手法では入力 x をそのままベースネットワークに送り、ルーティングネットワークには次元を合わせるための 1 層の全結合層を通して送るようにした。

ルーティングネットワークは、タスクの埋め込みベクトルとしてタスク id の onehot ベクトルを入力するため、タスク毎に使用するパラメータが分かれており、破滅的忘却が生じる可能性がないため SM と同様である。

LG と異なりモデルの切り替えがなく常にソフトなモジュール選択を行うため、モデルの更新は強化学習の更新アルゴリズムのネットワーク更新時にそのまま行う。

第5章　評価実験

5.1　実験環境

図 5.1: Freeway のプレイ画面

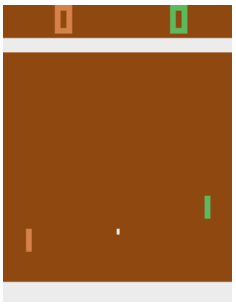

図 5.2: Pong のプレイ画面

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 通りの入力が可能な OpenAI Gym [13] の Atari 環境で、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 本体の 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。

5.1.1　Freeway

Freeway は図 5.1 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。

5.1.2 Pong

Pong は図 5.2 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。

5.2 実験設定

表 5.1: 各実験の task1 から task3（task3 は task1 と同一）

	task1	task2	task3
実験 1	Freeway	Freeway-ram	Freeway
実験 2	Freeway-ram	Freeway-ram-shift	Freeway-ram
実験 3	Freeway	Pong	Freeway

提案手法を評価するために、同じ環境で異なるモダリティとして画像入力の Freeway と状態入力の Freeway-ram、同じモダリティの状態入力でわずかに異なる環境として状態入力の Freeway-ram と行動を変形させた Freewah-ram-shift、同じモダリティの画像入力で異なる環境として Freeway と Pong、のそれぞれを順番に学習する実験を行った。各実験のタスク設定を表 5.1 に示す。それぞれ task1、task2、task3 の 3 つのタスクを順番に学習を行う継続的学習のタスク設定であり、task1 と task3 を同じにして破滅的忘却が生じていないかも確認する。各実験は、Learn to Grow（LG）ベースの提案手法と SoftModule（SM）ベースの提案手法の 2 つと、比較のために性能の下限として各層のモジュールを 1 つのままにする、図 4.1 の基本モデルのまま学習を行うベースラインの 3 つの手法で、学習アルゴリズムはいずれのタスクも学習可能な Double Deep Q-Network (DDQN) [25] を用いて学習を行う。SoftModule の論文では連続行動空間のタスクで評価していたため Soft Actor Critic（SAC）[39] が用いられていたが、Atari 環境は離散行動空間であり、SAC を離散行動空間に適用した SAC-Discrete [40] では Freeway や Pong を学習することは出来ていないため、モデル構造が多少変化しても学習することが出来る DDQN を採用した。

実験 1 では、同じ環境に対して画像入力と状態入力の両方を含む連続タスクで、有効な中間表現を獲得して学習が促進されるかを確認する。ベースラインでは、各層のモジュールは 1 つのままなので、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、同じ入出力形式でわずかに異なる環境を含む連続タスクで、変形された行動を出力する出力層で適切にモジュールの選択が行われ、中間層では表現が共有されるかを確認する。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、異なる環境を含む連続タスクで適切にモジュールを選択して破滅的忘却が生じないかを確認する。ベースラインでは、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。

5.2.1 モデルの構成

モデルは、図 4.1 のように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。ベースラインは基本モデルのまま、LG ベースの提案手法では各層に適切にモジュールを増やしていき、SM ベースの提案手法では各層に決められた数のモジュールを用意して学習を行う。入力は 4 チャンネル、出力は 32 チャンネル、カーネルサイズは 8×8、ストライドは 4 の畳み込み層を Conv2d(4, 32, 8, 4)、入力の次元が 512、出力の次元が 256 の全結合層を Linear(512, 256) のように表すとして、各層、モジュールの設定を表 5.2 に示す。LG ベースの提案手法での reuse や new モジュール、SM ベースの提案手法でのモジュールは、それぞれ基本モデルの層と同じサイズである。活性化関数は全て ReLU を用いた。LG ベースの提案手法のモデルは学習中に各層のモジュール数が変化するため、例として実験 1 の task2 を学習中のモデルのベースネットワークを図 5.3、ルーティングネットワークを図 5.4 に示す。SM ベースの提案手法のモデルは、本研究では各層のモジュール数を $n = 2$ としており、ベースネットワークを図 5.5、ルーティングネットワークを図 5.6 に示す。図 5.3–5.6 の実線は実験 1 の task2 である Freeway-ram を学習中に使用する経路を表す。

5.2.2 パラメータ設定

表 5.3 に実験に用いたパラメータ設定を示す。

表 5.2: 各層、モジュールの設定

層/モジュール	設定
Conv1	Conv2d(4, 32, 8, 4)
Conv1-adapter	Conv2d(32, 32, 1, 1)
Conv2	Conv2d(32, 64, 4, 2)
Conv2-adapter	Conv2d(64, 64, 1, 1)
Conv3	Conv2d(64, 64, 3, 1)
Conv3-adapter	Conv2d(64, 64, 1, 1)
fc1 (img)	Linear(3136, 512)
fc1 (img)-adapter	Linear(3136, 196), Linear(196, 512)
fc1 (ram)	Linear(128, 512)
fc1 (ram)-adapter	Linear(128, 8), Linear(8, 512)
fc2	Linear(512, 256)
fc2-adapter	Linear(512, 32), Linear(32, 256)
fc3	Linear(256, 128)
fc3-adapter	Linear(256, 16), Linear(16, 128)
fc4 (Freeway)	Linear(128, 3)
fc4 (Freeway)-adapter	Linear(128, 8), Linear(8, 3)
fc4 (Pong)	Linear(128, 6)
fc4 (Pong)-adapter	Linear(128,8), Linear(8, 3)

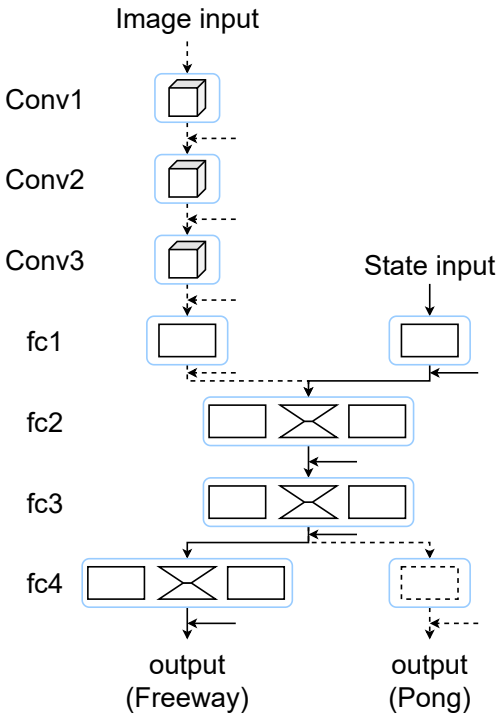

図 5.3: 学習中の提案手法（LG ベース）のベースネットワーク

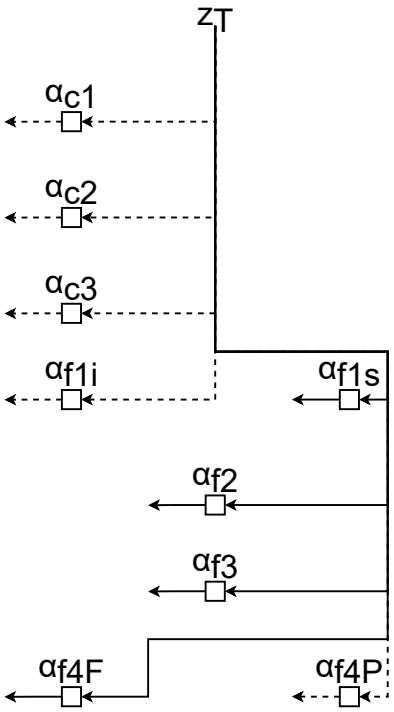

図 5.4: 学習中の提案手法（LG ベース）のルーティングネットワーク

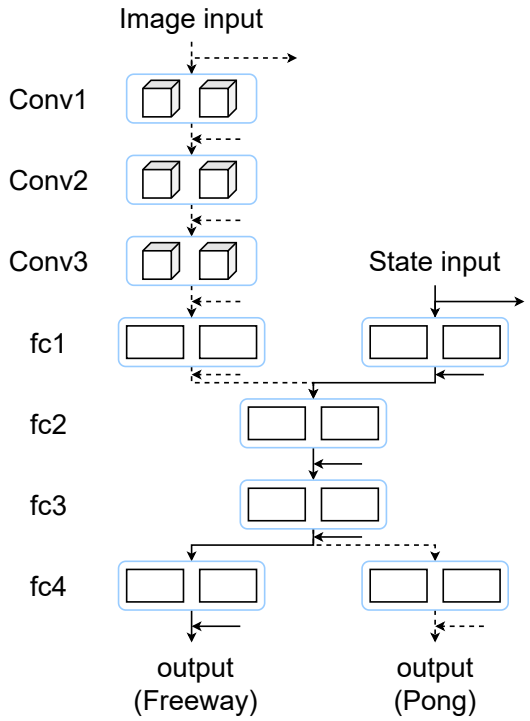

図 5.5: 提案手法（SM ベース）のベースネットワーク

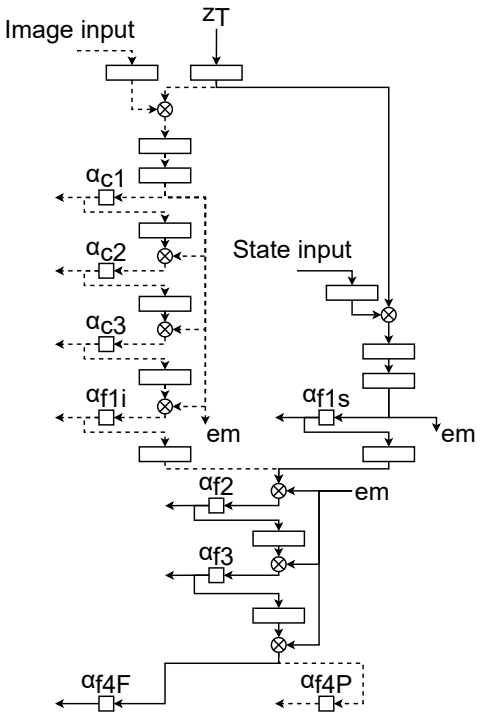

図 5.6: 提案手法（SM ベース）のルーティングネットワーク

表 5.3: 実験に用いたパラメータ設定

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（LG ベース）	1e-5
L2 ペナルティ係数 λ（LG ベース）	0
割引率 γ	0.99
構造探索停止のステップ数（LG ベース）	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

5.3 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。図中ではベースラインを baseline、Learn to Grow ベースの提案手法を prop. (LG)、SoftModule ベースの提案手法を prop. (SM) 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 ではベースラインと LG ベースの提案手法は同じ基本モデルで学習を行うためまとめて表記している。

5.3.1 実験 1: Freeway 画像-状態-画像

結果を図 5.7、5.8、5.9 に示す。図 5.7、5.9 より、提案手法はどちらも 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げている。一方で、ベースラインでは task2 の状態を挟んだことにより、破滅的忘却が生じて task1 の学習に近い 3.5M step ほどで 30 付近の報酬を得る学習となっている。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 5.8 では、LG ベースの提案手法はばらつきが大きいものの 3M step ほどで 30 近くの報酬を得ており提案手法の方が、5M step でも 28 近くの報酬しか得られていないベースラインよりや、図 5.10 より同じ Freeway-ram である実験 2 task1 のランダムな初期化後の 4.5M step ほどで 30 付近の報酬を得ているベースラインよりも学習が早い。ランダムな初期化後よりも Freeway 学習後の方が Freeway-ram のベースラインの学習性能が低下しており、手法の工夫をせずに連続して学習を行うベースラインでは、過去のタスクの学習により現在のタスクに対する性能が低下する現象が生じている。提案手法のモデルの fc2-3 層では、LG ベースでは task1 で学習したモジュールを再利用 (reuse) したものが多くなっており、SM ベースでは task1 とルーティングパラメータが近くなっていることから、入力の形式がゲーム画面とゲーム内のメモリと異なっていても同じゲームであるため、task1 で学習したモジュールを有効に使うことができている。

5.3.2 実験 2: Freeway 状態-状態 (行動変形)-状態

結果を図 5.10、5.11、5.12 に示す。図 5.10、5.12 よりベースライン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 5.11、5.12 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。

LG ベースの提案手法が task2 で選択したモジュールは、fc2-3 層ではほとんどが reuse モジュール、fc4 層では adapt モジュールであり、出力に変化を加えたタスクにおいて出力層のモジュールだけを追加しているため、適切な層のパラメータを追加できている。

5.3.3 実験 3: Freeway 画像-Pong 画像-Freeway 画像

結果を図 5.13、5.14、5.15 に示す。実験 3 の task1 は実験 1 の task1 と同じなので、図 5.13 は図 5.7 と同じである。図 5.7、5.15 より、提案手法ではすぐに 30 以上の報酬を得ており、破滅的忘却を防げている。一方で、ベースラインでは task1 の学習結果とほとんど同じ学習で 3.5M step ほどで 30 付近の報酬を得ており、破滅的忘却が生じている。

図 5.14 では、LG ベースの提案手法はばらつきが大きいがいずれも 1M step ほどで 20 付近の報酬を得る似た学習結果になっている。LG ベースの提案手法だけばらつきが大きいのは、LG ベースの提案手法が 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 で学習したモジュールを利用することができず、構造探索で適切なモジュール選択が出来ない場合が多いためである。SM ベースの提案手法も同様に task1 で学習したモジュールを利用することができないが、常にソフトなモジュール選択を行うため、LG ベースの提案手法と異なり不安定にならずに学習することが出来たと考えられる。

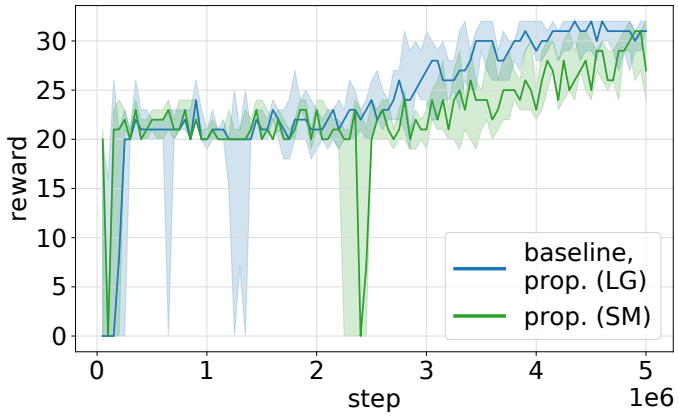

図 5.7: 実験 1 task1: Freeway 学習結果

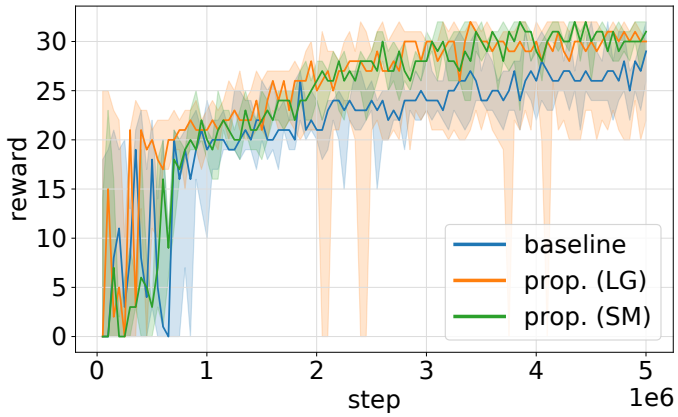

図 5.8: 実験 1 task2: Freeway-ram 学習結果

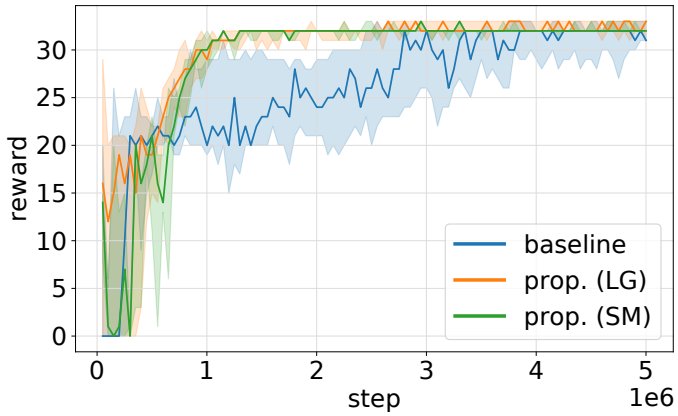

図 5.9: 実験 1 task3: Freeway 学習結果

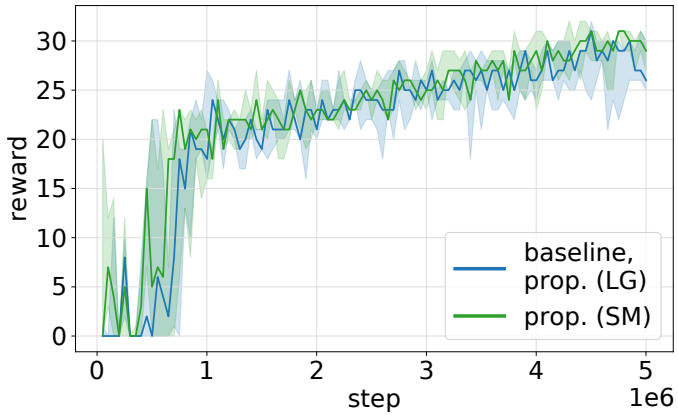

図 5.10: 実験 2 task1: Freeway-ram 学習結果

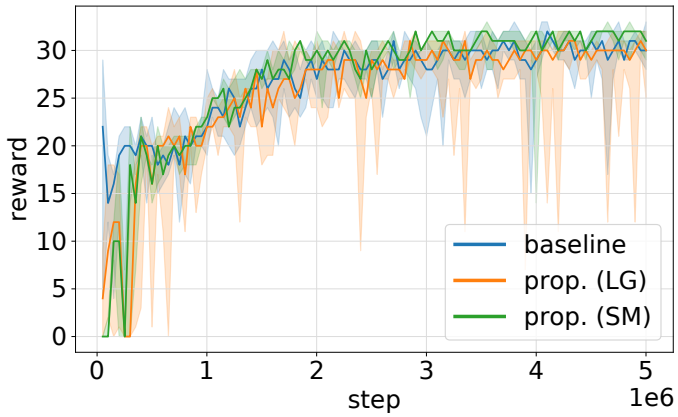

図 5.11: 実験 2 task2: Freeway-ram-shift 学習結果

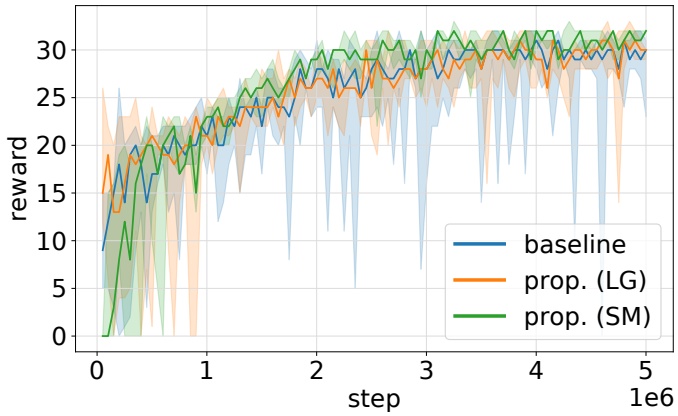

図 5.12: 実験 2 task3: Freeway-ram 学習結果

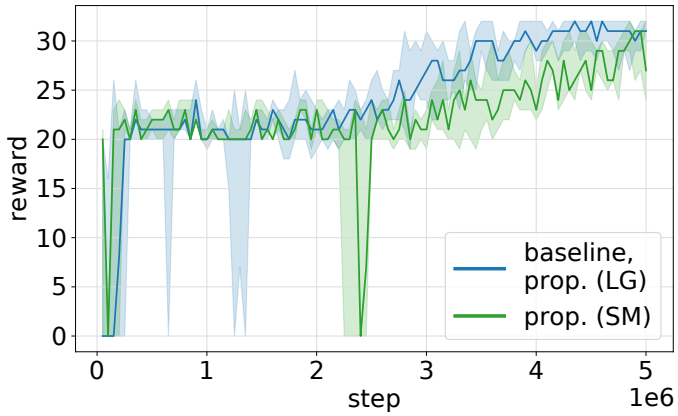

図 5.13: 実験 3 task1: Freeway 学習結果

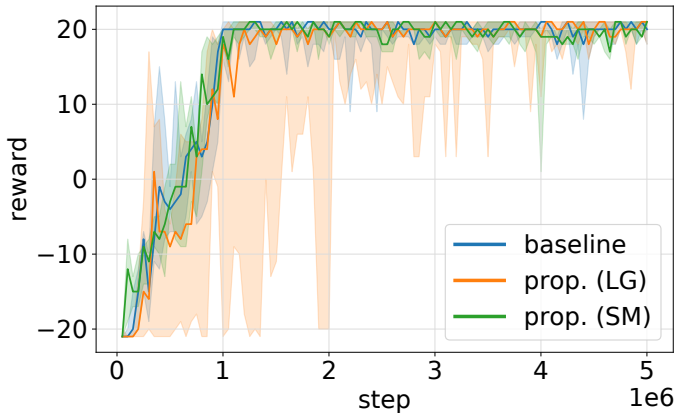

図 5.14: 実験 3 task2: Pong 画像入力の学習結果

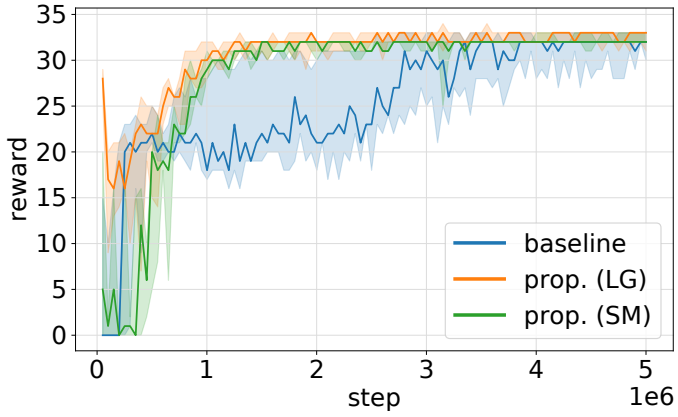

図 5.15: 実験 3 task3: Freeway 画像入力の学習結果

第6章　考察

初めに、各層のモジュールが 1 つのベースラインの学習結果に注目する。図 5.7、5.9、5.15 より、Freeway の画像入力で最適解に近い 30 付近の報酬を得たのは、実験 1/3 の task1 は約 3.5M step、実験 1/3 の task3 は約 3M step であり、局所最適解の 23 以上の報酬を得られるようになるのは、実験 1/3 の task1 は約 2.5M step、実験 1 の task3 は約 1.5M step、実験 3 の task3 は約 2.5M step と、実験 3 では実験 1 よりも破滅的忘却が生じている。task2 が Pong である実験 3 の方がより異なるタスクであり必要な表現が異なるため、より忘却が生じている。

図 5.10、5.11、5.12 より、task1-3 が Freeway の状態入力で task2 は出力層だけで対応することが出来る行動変形である実験 2 では、30 付近の報酬を得たのは task1 は約 4.5M step、task2 は約 2M step、task3 は約 3M step と、過去のタスクの学習により現在のタスクに対する性能が向上しているが、task3 では task1 の結果と異なり 1M step 以降でも常に上方向の行動を行えば得られる局所最適解の報酬である 23 付近を最小値では何度も下回っている。これは、実験 2 は task1 と task2 がよく似たタスクであり、ほどんどの場合は破滅的忘却が生じないが、生じた場合は報酬が 23 を何度も下回る不安定な学習になる設定である考えられる。

図 5.10、5.8 より、Freeway の状態入力の結果を比較すると、ランダムな初期値から学習した実験 2 の task1 では約 2M step で 23 付近、約 4.5M step で 30 付近の報酬を得るような学習に対して、画像入力の学習後から学習した実験 1 の task2 では約 2.5M step で 23 付近、5M step でも 29 以下の報酬を得ており、性能が低下している。Freeway の画像入力は、特に工夫をしないで学習を行うと、状態入力の学習に悪い影響を与えるタスクとなる。

次に、各層のモジュールを複数持つ LG ベースと SM ベースの提案手法の学習結果に注目する。どちらの手法も 5.3 節より破滅的忘却を防げているが、各実験の task2 の図 5.8、5.11、5.14 より LG ベースの提案手法は、ベースラインや SM ベースの提案手法と比較して時々大きく報酬が低下しており学習が安定していない。これは、学習中にモジュールのソフトな選択とハードな選択を交互に切り替えて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。SM ベースの提案手法は、図 5.10 より Freeway の状態入力ではベースラインや LG ベースの提案手法とほぼ同じ学習結果だが、図 5.7 より Freeway の画像入力では報酬が 20 付近から上がって 30 付近になるのがやや遅くなっている。これは、SM ベースの提案手法では初めから各層にモジュールが $n = 2$ 個と複数存在するため、モジュールが 1 個の基本モデルと比べてモデルが大きく、学習が遅くなってしまったためと考えられる。

本稿では、複数のタスクを扱う手法の図 2.1 のような分類において、タスク毎にモジュールを増やしていかない手法として SoftModule（SM）、タスクに応じてモジュールを増やしていく手法として Learn to Grow（LG）を用いた。タスク毎にモジュールを増やしていく手法は、破滅的忘却が生じ

ないが、計算コストがタスク数に応じて非常に高くなり実験コストが大きく、より様々なタスクを扱うという目標にも適していないため本実験では評価していない。LG ベースの提案手法は、パラメータ量を比較的抑えることが出来るが、モジュールを新しく増やすかどうかの学習でモデルが大きく変化するため学習が不安定になり、SM ベースの提案手法は、学習が安定するが初めからモジュール数が固定であるため、連続タスクの初めの方のタスクに対してはモデルが大きく学習が遅くなることが分かった。このため、強化学習の複数のタスクを扱う手法では、パラメータ量による計算コストと扱うことのできるタスクの数のトレードオフだけでなく、モデルの構造や学習の安定性を踏まえた改善が必要となる。

第7章　結論

7.1　まとめ

本稿では、複数のタスクを扱う手法の、扱えるタスクが似たようなタスクである狭い範囲に限定されているという課題に対して、様々な状況を含む広い範囲のタスクを扱えるようにするという既存の方針だけでなく、様々なモダリティのタスクを扱えるようにする必要があることを指摘した。複数のモダリティとして、強化学習で良く用いられている画像と状態の 2 つを扱えるモデルを、既存の複数のタスクを扱う手法である Learn to Grow と SoftModule のそれぞれで学習するモデルを提案した。画像入力と状態入力のどちらも扱うことが出来る OpenAI Gym の Atari 環境のゲームを用いて、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、複数のタスクを扱う手法の課題である破滅的忘却を防ぐことができており、過去のタスクの学習により現在のタスクに対する性能が向上する効率的な学習が行えることを確認した。

7.2　今後の課題

今後の課題としては、学習手法の改善が挙げられる。既存の複数のタスクを扱う手法は教師有り学習を対象にしたものが多く、Leanr to Grow のように強化学習に適用すると学習がかなり不安定になってしまう。SoftModule のような強化学習タスクを扱う手法は、強化学習タスクに対して学習が比較的安定するが、タスクに対する追加するパラメータの量をタスクによらず決定する手法が多く、計算コストの増大やモデルの表現力の不足といった問題が生じる。強化学習タスクに対しても過去のタスクとの関係から適切パラメータを追加して、安定した学習が可能な必要であり、今後の研究に期待したい。また、本稿では複数のモダリティのタスクを含む連続タスクとして、Atari 環境の 3 つの連続タスクで評価していたが、Atari 環境の連続タスクとしては扱う環境が少なく連続タスクとしても短いため、既存の研究と同様に 10 以上の様々ゲームを含んだ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] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. Vol. 24 of *Psychology of Learning and Motivation*, pp. 109–165. Academic Press, 1989.

[2] B. Pfülb, A. Gepperth, S. Abdullah, and A. Kilian. Catastrophic forgetting: still a problem for dnns. In *Artificial Neural Networks and Machine Learning (ICANN)*, Vol. 487–497, 2018.

[3] Mark Bishop Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, USA, 1994. UMI Order No. GAX95-06083.

[4] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[5] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80 of *Proceedings of Machine Learning Research*, pp. 2497–2506. PMLR, 10–15 Jul 2018.

[6] Rich Caruana. Multitask learning. In Sebastian Thrun and Lorien Y. Pratt, editors, *Learning to Learn*, pp. 95–133. Springer, 1998.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. Vol. 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[8] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, Vol. 114, No. 13, pp. 3521–3526, 2017.

[9] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2019.

[10] Paul Pu Liang, Peter Wu, Liu Ziyin, Louis-Philippe Morency, and Ruslan Salakhutdinov. *Cross-Modal Generalization: Learning in Low Resource Modalities via Meta-Alignment*, pp. 2680–2689. Association for Computing Machinery, New York, NY, USA, 2021.

[11] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3925–3934. PMLR, 2019.

[12] Ruihan Yang, Huazhe Xu, YI WU, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, Vol. 33, pp. 4767–4777. Curran Associates, Inc., 2020.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. cite arxiv:1606.01540.

[14] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, Vol. Vol. 47, pp. 253–279, 2012. cite arxiv:1207.4708.

[15] Richard S. Sutton. Learning to predict by the methods of temporal differences. In *MACHINE LEARNING*, pp. 9–44, 1988.

[16] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report TR 166, Cambridge University Engineering Department, Cambridge, England, 1994.

[17] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 2013. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, Vol. 86, pp. 2278–2324, 1998.

[20] L. J. Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, January 1993.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, Vol. 518, No. 7540, pp. 529–533, February 2015.

[22] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

[23] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In Michael Mozer, Paul Smolensky, David Touretzky, Jeffrey Elman, and Andreas Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School*, pp. 255–263. Lawrence Erlbaum, 1993.

[24] Hado Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, Vol. 23. Curran Associates, Inc., 2010.

[25] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. 2015. cite arxiv:1509.06461Comment: AAAI 2016.

[26] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[27] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[28] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[29] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1995–2003. JMLR.org, 2016.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[31] Steven Kapturowski, Georg Ostrovski, John Quan, Rémi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[32] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 06–11 Aug 2017.

[33] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *AAAI*, pp. 3215–3222. AAAI Press, 2018.

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

[35] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, Vol. 32 of *Proceedings of Machine Learning Research*, pp. 387–395, Bejing, China, 22–24 Jun 2014. PMLR.

[36] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

[37] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 10–15 Jul 2018.

[38] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1352–1361. JMLR.org, 2017.

[39] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. Vol. 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[40] Petros Christodoulou. Soft actor-critic for discrete action settings. Vol. abs/1910.07207, 2019.

[41] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *J. Mach. Learn. Res.*, Vol. 17, pp. 81:1–81:32, 2016.

[42] Nilesh Tripuraneni, Michael Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and

H. Lin, editors, *Advances in Neural Information Processing Systems*, Vol. 33, pp. 7852–7862. Curran Associates, Inc., 2020.

[43] Sebastian Ruder. An overview of multi-task learning in deep neural networks. Vol. abs/1706.05098, 2017.

[44] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48 of *Proceedings of Machine Learning Research*, pp. 1842–1850, New York, New York, USA, 20–22 Jun 2016. PMLR.

[45] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. Vol. abs/1803.02999, 2018.

[46] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. Vol. abs/1606.04671, 2016.

[47] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 06–11 Aug 2017.

[48] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. Vol. abs/1611.01796, 2016.

[49] Prasoon Goyal, Scott Niekum, and Raymond J. Mooney. Using natural language for reward shaping in reinforcement learning. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 2385–2391. ijcai.org, 2019.

[50] Faraaz Nadeem. Learning from musical feedback with sonic the hedgehog. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, pp. 476–483, 2021.

[51] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

[52] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[53] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., 2017.

[54] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[55] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[56] Youssef Mroueh, Etienne Marcheret, and Vaibhava Goel. Deep multimodal learning for audio-visual speech recognition. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2130–2134, 2015.

[57] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 689–696, Madison, WI, USA, 2011. Omnipress.

[58] Carina Silberer and Mirella Lapata. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 721–732, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[59] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *arXiv preprint arXiv:1705.09406*, 2017.

対外発表

川島 丸生, 伊庭 斉志. 複数の入出力サイズを扱う継続的強化学習手法. 第 25 回ゲームプログラミングワークショップ, pp. 161-168, 2020. （研究奨励賞受賞）

謝辞

本研究を進めるに当たり、様々な方にお世話になりました。

指導教員である伊庭斉志教授には、ミーティングなどを通じて、自分では気づきにくい視点からのアドバイスやご助言をいただきました。また、研究テーマを自由にやらせていただき、卒業論文のテーマをさらに進めた修士論文となり、感謝しています。

鶴岡研究室の鶴岡慶雅教授には、ミーティングへの参加を許可していただいただけでなく、様々なアドバイスもしていただき、本当にありがとうございました。

同期や先輩、後輩の学生の皆さんにも、ミーティングなどで様々な意見をいただき、研究分野に関する知識の共有や議論が出来て、研究を進めることが出来ました。伊庭研究室の先輩である宮崎広夢先輩や鶴岡研究室の先輩である李凌寒先輩には、特に発表の形式などのアドバイスをいただき、鶴岡研究室の同期である橋本大世君や綿引隼人君には、研究テーマの論旨や手法について様々な指摘をいただきました。特に、綿引隼人君には、研究実験のための管理ツールを教えて頂いたり、研究実験のための基本的なアルゴリズムの実装を参考にさせていただき、本当に助かりました。皆様方の協力があったからこそ、本研究を進めていくことが出来たのだと感じております。本当にありがとうございました。