修　　士　　論　　文

モデルベース深層強化学習における タスク依存の中間表現を用いた 環境遷移モデルの学習

Learning Environment Models Based on Task-Specific Representations in Model-Based Deep Reinforcement Learning

指導教員　　鶴岡 慶雅 教授

東京大学大学院工学系研究科
電気系工学専攻

氏 名　　**37176488 水谷 陽太**

提 出 日　　平成 31 年 1 月 31 日

概 要

実空間やビデオゲームなどの複雑な環境においてタスクの達成に最適な方策を得るために，深層学習技術を用いて強化学習を行う深層強化学習に関する研究が注目を集めているが，有用な方策を得るためには多くの場合において大量の試行が必要となり，サンプル効率の悪さが問題となっている．そこで近年では，環境モデルを学習して探索を行うことで，長期的な戦略に基づく行動を可能とし，サンプル効率を改善したモデルベース強化学習が盛んに研究されている．しかし高次元の観測データの遷移規則を学習する場合には，計算量の増大により学習が長時間化してしまう傾向にある．いくつかの研究では観測を低次元の中間表現に変換し，中間表現の空間で環境モデルを学習することで計算量を削減することを試みているが，多くの場合に中間表現の事前学習が必要であり，方策により環境から得られる観測の分布が変化する強化学習において問題となることが懸念される．本研究では計算効率及び学習効率を上げるために，タスクを達成するのに適した低次元の中間表現およびその中間表現を用いた環境遷移モデルを方策と同時に学習する手法を提案する．また倉庫番ゲームによる評価実験により，提案アーキテクチャが実際に短時間の学習で性能の良い方策を効率よく学習可能であることを確認する．

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

近年の計算機性能の向上や深層学習技術の発展に伴い，機械学習，人工知能分野に関する研究が注目を集めている．深層学習は，画像認識や自然言語処理などの分野でめざましい成果をあげているが，近年では深層学習を用いてゲームのコンピュータプレイヤを実現する研究も行われている．ゲーム AI 研究の目的の一つに，現実世界における問題解決への応用が挙げられる．ゲームは制約や報酬などの条件が明確に定められ，やり直しが用意で，性能の向上を客観的に評価することが容易であるため，人工知能のテストベットに適している．設定が複雑な現実の問題への適用を考えた場合，特定のゲームに特化した手法よりもより汎用的に適用できる手法のほうがより優れていると考えられる．また，ロボットの制御などに応用する場合，リアルタイムに計算できるように，計算コストを抑えることも重要になる．古くは，リバーシ [1] などのボードゲームの AI が研究されていたが，これらはゲームの状態遷移規則が事前に判明していることを前提としており，現在の状態から仮想的に状態遷移を繰り返すことによる探索などを行っている．そのため状態遷移規則が未知であるような現実の問題への応用が難しく，また探索には比較的多くの計算リソースが必要であった．近年ではより現実に近い設定として，ビデオゲームの AI に関する研究が盛んに行われている．2013 年に Mnih らが発表した Deep Q-Network (DQN) [2] を用いたエージェントは，ゲーム画面を直接入力して学習を行うことができる．DQN はビデオゲーム機の Atari 2600 における幾つかのゲームにおいて，人間のトッププレイヤーのスコアを上回る結果を出すことに成功した．また従来の人工知能とは異なり，ゲームのルールを事前に教えられることなく，自らの試行により学習を行う．DQN には汎用性があり，Atari の多くのゲームに対して同じ構造，パラメータを用いて学習を行うことが可能であることが示された．DQN は深層学習と強化学習を組み合わせた手法を用いており，DQN の改良を行う研究が近年盛んに行われている [3, 4].

強化学習とは機械学習の一種であり，ある環境のもとにおかれたエージェントが，現在の状態を観測し，行動に対する報酬を環境から得ながら，試行錯誤を通して報酬を最大化するための行動戦略を学習するための枠組みである [5]．既に，制御工学分野等への応用が研究されている [6, 7].

DQN やその発展である Asynchronous Advantage Actor-Critic [8] では，モデルフリーな手法で学習を行っている．ここでいうモデルとはエージェントが行動する環境のモデルを意味する．すなわち，環境に対して行動を行った際，環境がどのように変化してどのような報酬が得られるかという情報である．モデルフリーな手法はこれらの情報を直接的には使用しない．DQN 等は，ある状態において特定の行動を取った場合，どの程度報酬を受け取ることができるかという情報を経験から学習する．完全な学習が行えると仮定すると，このような手法でも最適な方策を学習することができるが，ビデオゲームのような複雑なモデルにおいて，各状態と行動における価値を完全に学習することは不可能に近い．限りあるリソースで学習を行うため，DQN 等はニューラルネットワークによる関数近似を用いている．学習が不完全であることにより，モデルフリーの従来手法では，将来の状態を見据えて長期的な戦略を立てることを苦手とする傾向にある．また，状態毎に最適な方策を学ぶ必要があるため，学習の収束に非常に多くの経験を必要とする．これは現実の問題に適用する際に無視できない問題となる．

これに対して，環境の状態遷移を用いて方策を決定する手法をモデルベースな手法と呼ぶ．環境のモデルを利用し，実際に行動する前に仮想的に状態遷移を行い，探索をすることで，先の状態を見据えて長期的な戦略を立てることが可能となる．Atari 2600 において，ALE と呼ばれるエミュレータを用いることで環境のモデルを得て，モンテカルロ木探索を行った研究が存在する [9]．強化学習の分野においては，囲碁について，ゲームの状態遷移を利用し，人間のトッププレイヤーに勝る強さの AI を実現することに成功した研究が存在する [10]．これらの手法は事前に環境の状態遷移が全て判明している必要があり，現実の複雑な問題に適用することは難しい．そこで，事前に環境のモデルが判明していない状態から，環境のモデルを学習する研究が行われている [11]．このように強化学習と環境モデルの学習を組み合わせてよりよい方策を得るための手法をモデルベース強化学習と呼ぶ．モデルベース強化学習は，環境モデルを用いた探索により，比較的学習のサンプル効率が良い [12]．

しかし環境のモデルを直接学習することは難しく，また不完全に学習された環境で探索を行うと逆に性能を悪化させるおそれがある．そのため，Predictron（Silver ら，2017）では，環境の遷移を直接学ぶのではなく，人の手で事前に設計した特徴量の遷移を学習する [13]．これに対し，Value Prediction Network（Oh ら，2017）は，特徴量を抽出するネットワークを含めて学習を行い，木探索によって期待報酬の最大値を求める [14]．

環境のモデルが正確である場合は，木探索を深くして，末尾の状態を用いることで，より先の状態を考慮した方策を得ることができるが，環境のモデルが不完全である場合は，探索を深くすることによりモデルの誤差が蓄積されるおそれがある．実際に，不完全なモデルで木探索を行うとモデ

ルフリーな手法よりも性能が悪化する実験結果が報告されている [15]．その為，RL-HLP（亀甲ら，2017） [16] や I2A（Weber ら，2017） [15] では，探索中に現れた途中の状態全ての軌跡をニューラルネットワークに入力することで方策の決定を行う．これにより，予測の信頼度も含めて環境モデルから得られた予測情報の最適な扱い方をニューラルネットワークが学習することでよりよい方策を得ることができる．

深層強化学習の分野では，環境からの観測としてゲーム画面のような高次元のデータを用いることが多い．従って観測を元にそのまま環境モデルを定義してしまうと高次元の入力から高次元の出力を予測するニューラルネットワークを学習する必要があり，計算量の増大を招く．そこで観測を低次元の中間表現に抽象化した上で環境モデルの学習を行う手法が提案されているが [17, 18]，中間表現の事前学習が必要であるなどの課題が存在する．

本稿では，モデルベース強化学習のための新しいアーキテクチャを提案する．提案するアーキテクチャは，畳み込みニューラルネットワークにより得た中間表現を環境モデルを用いた探索により得られた特徴量と結合して方策を学習することにより，タスク達成に適した低次元の中間表現を同時に学習する．加えて，分散強化学習の手法である Importance Weighted Actor-Learner Architecture (IMPALA) [19] の学習データの特性を利用し，中間表現の空間での環境モデルの学習を方策と同時に効率よく行うことができる．これらの工夫により，高次元の観測を扱う場合でも比較的小さな計算コストで高効率に方策を得ることを可能とした．倉庫番ゲームによる評価実験にて，提案したアーキテクチャが短時間の学習で性能の良い方策を得ることができることを確認した．

1.2 本研究の目的と貢献

本研究の目的は，マルコフ決定過程における強化学習において，高次元の入力を観測として用いる場合の性能向上である．ここでの性能とは，学習が収束した際に得られる方策の精度，学習のサンプル効率および学習時やテスト時の計算時間の削減が挙げられる．これらを高めつつ，より一般的で複数のタスクに適用可能な手法を模索する．その上で，サンプル効率や長期的な計画を立てる能力に優れると考えられているモデルベース強化学習を使用する．しかし先に挙げた要素のうち，学習時やテスト時の計算時間という点において，モデルベース強化学習は課題を残している．本研究ではその問題点を改善するための新しいアーキテクチャを提案する．

高次元の観測データにおける状態遷移をそのまま環境モデルとして学習すると，環境モデルの学習時および利用時の計算コストが大きくなってしまう．この問題を解決するため，高次元の観測データを一度低次元な中間表現に抽象化した後，その中間表現における状態遷移を学習する研究が行わ

れている [16, 17, 18, 20]．しかし既存の研究では，中間表現の事前学習が必要であり，方策の改善に伴う観測データの分布変化に対応できないなどの欠点が存在した．

本研究で提案するアーキテクチャは，上記で述べたような観測を低次元の中間表現に抽象化するモデルベース強化学習の一種である．既存手法と異なる主な貢献として，モデルを表現する適切な中間表現の獲得を，モデルを用いた方策の獲得と同時に行うことができるという点が挙げられる．すなわち，既存手法においては，適切な中間表現を得るための学習を行った後に，モデルベース強化学習を行い方策を獲得していたが，本稿で提案するアーキテクチャは，それらの学習を同時に行うことで，強化学習における入力分布の変化に対応することを可能とした．同時学習を実現するために，提案アーキテクチャは環境モデルを用いた予測により得られる特徴量に，予測を用いないモデルフリーな特徴量を結合して方策決定を行い，その学習で得られたモデルフリーな特徴量を中間表現として利用する．

1.3 本稿の構成

本稿の構成を以下に述べる．

第 2 章では，強化学習の基礎知識について述べる．第 2.1 節では，強化学習の問題設定や基本となる学習アーキテクチャについて述べ，第 2.2 節では，本稿で用いた分散アーキテクチャについて説明する．

第 3 章では，モデルベース強化学習について述べる．モデルベース強化学習が抱える課題と，それを解決すべく提案された既存手法を紹介，比較する．

第 4 章では，本稿で提案するモデルベース強化学習の新しいアーキテクチャについて述べる．第 4.1 節では，アーキテクチャの概要および各モジュールの説明を行う．第 4.2 節では，倉庫番ゲームにおける評価実験の内容およびその結果，考察を述べる．

最後に第 5 章では，以上の内容を踏まえて，本稿で提案したアーキテクチャについてまとめた．

第2章　前提知識

2.1　強化学習

強化学習 [5] とは，マルコフ決定過程（Markov Decision Process, MDP）の環境において，最適な方策を得るための機械学習の枠組みである．エージェントはある環境下において行動をとり，環境から行動に対する報酬と次の状態の観測を得る．エージェントが行動を取ることにより状態は確率的に遷移し，遷移した状態に対応する観測と報酬が与えられる．これを繰り返すことで，エージェントは累積報酬を最も大きくする方策を学習する．

時刻 t における状態 s_t において，エージェントが行動 a_t を取った場合，環境は状態遷移関数 $T(s_{t+1}|s_t, a_t)$ で表される確率に従って状態 s_{t+1} へと遷移する．この際エージェントは報酬関数 $r(s_t, a_t, s_{t+1})$ によって規定される報酬を得る．エージェントはこれらの確率分布を事前に知ることはできず，試行錯誤の中で最適な方策を学習していく．

最適な方策とは，各時刻 t で得られる報酬を r_t とするとき，時刻 t において，$0 < \gamma < 1$ を満たすパラメータ γ を用いて

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$$

という R_t の期待値を最大化するものである．この際 γ を割引率と呼ぶ．

このようにして得られた方策を $\pi(a_t|s_t)$ とする．方策関数 π は状態 s_t に対して行動 a_t を行う確率を表す．

ここまで状態 s_t と観測を同一のものとして扱ってきたが，毎周期状態の完全な観測が得られない設定の問題も多く存在し，それらは部分観測マルコフ決定過程（Partially Observed Markov Decision Process, POMDP）と呼ばれる．この場合 s_t と観測 o_t を区別し，POMDP における最適方策を得るためには過去の観測の記憶が必要となる．本稿で提案するアーキテクチャは常に状態を完全に観測可能である環境のみを扱うこととしているため，現在の状態と観測を同一視し，現在の観測のみから方策を決定する．POMDP への適用可能性については第 5.2 節で述べる．

2.1.1 Q 学習

時刻 t におけるある特定の状態 s_t において，行動 a_t を取った場合の価値のことを Q 値と呼び，$Q^*(s_t, a_t)$ で表す．Q 値は，s_t において a_t を行った場合の報酬 r_{t+1} に，その後最適な行動を取り続けた場合の R_{t+1} を足した値の期待値である．すなわち，累積報酬を最大化する最適な方策を π^* と置くと，

$$Q^*(s_t, a_t) = \mathbb{E}\left[r_t + \sum_{t'=t+1} \gamma^{t'-t} r_{t'}\right]$$

となる．ただし，$t' \geq t+1$ において，$s_{t'}$ から $s_{t'+1}$ に遷移する際の行動は方策 π^* に従って決定するとする．

ここで，Q^* は以下の式を満たす．

$$Q^*(s_t, a_t) = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})\right]$$

この方程式をベルマン方程式と呼ぶ．

Q 学習とは，エージェントにとって未知である Q 値を試行の中で徐々に推定していくことで，最適な方策を学ぶ手法である．

より具体的には，各状態と行動の組について，$Q^*(s_t, a_t)$ の推定値 $Q(s_t, a_t)$ を用意し，エージェントの経験から得られた s_t, a_t, s_{t+1} の組を用いて，Q がベルマン方程式を満たすように更新していく．

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)$$

この更新で最適方策を得るためには多数の s_t, a_t の組に対する試行により得られたデータが必要となる．例えば行動をランダムに取るエージェントの経験から得られたデータを使用する場合でも理論上は正しい Q 学習が可能である．しかしゲームなどの MDP 環境を考えると，ある程度有用な方策を用いて行動を行わないと十分な観測を経験できない場合が存在する．例えば序盤のステージをクリアすることでボスステージに挑むことができるようになるゲームの場合，ランダムエージェントが序盤ステージを攻略する可能性は限りなく低いことが考えられる．その場合，ランダムエージェントではボスステージの経験を得ることができない．よって，より効率よく学習データを集めるために，学習中の Q を用いた行動決定をすることで，ある程度よい方策に従って行動し，データを集める手法がよく用いられる．この場合，$\arg\max_{a_t} Q(s_t, a_t)$ に従って a_t を決定する．ただし，常に Q が最大である行動しか取らない場合は局所解に陥ってしまい，学習に必要な経験が得られなくなってしまうことが懸念される．そのため確率 ε でランダムに行動を選択し，$1-\varepsilon$ で Q に従った行動選択

をして経験を収集する手法がよく用いられる．これを ε-greedy 法と呼ぶ．なお，学習が終了した後のテスト時には単に Q が最大となる行動をとり続けることで最適な方策となる．

しかし全ての s と a の組において $Q(s,a)$ を保存する場合，状態数や行動の数が増えるに従い保存すべき値が増え，現実的なリソースでの計算が困難になる．そこで，$Q(s,a)$ の推定値を直接保持する代わりに，何らかの関数により $Q(s,a)$ を近似する手法が提案されている [21]．

2.1.2 Deep Q-Network (DQN)

Mnih らが 2013 年に提案した DQN (Deep Q-Network) は，Q 学習における Q 値の近似関数を，畳み込みニューラルネットワークを用いて表したものである [2, 22]．ゲーム画面を観測，ゲームのスコアを報酬，コントローラ入力をエージェントの行動とした学習の後，Atari 2600 の複数のゲームで人間のトッププレイヤーを上回るスコアを達成した．

DQN を用いたエージェントは観測として得たゲーム画面を，畳み込みニューラルネットワークに入力する．ニューラルネットワークの出力として，N_a 次元のベクトルを得る．ここで N_a は取り得る行動の種類数であり，ベクトルの n 次元目の値が n 種類目の行動 a に対応した Q 値を表す．すなわち，$Q(s_t,a_t;\theta)$ は，ニューラルネットワークのパラメータを θ として，ゲーム画面 s_t をニューラルネットワークに入力した際の，a_t に対応する出力によって表される．このように Q 値を近似するためのニューラルネットワークを Q-Network と呼ぶ．学習時には

$$L = \left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^-) - Q(s_t, a_t; \theta) \right)^2$$

を誤差関数として使用する．すなわち，現在の $Q(s_t,a_t;\theta)$ を実際に経験して得られた r_t を元に，$r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^-)$ に近づけるように学習を行う．ここで θ^- は数バッチ前の θ のコピーであり，学習時には θ のみを更新する．一定回数の学習毎に θ を θ^- にコピーする．この θ^- を用いて $Q(s_{t+1},a_{t+1})$ を計算するネットワークは target network と呼ばれる．

強化学習における学習データは，時系列に沿ってエージェントが行動した順に与えられるため，サンプル間に強い相関関係が存在し，学習に悪影響を与えやすい．そのため DQN では，得られた経験を replay memory と呼ばれるメモリ領域に保存しておき，学習を行う際には replay memory からランダムに経験をサンプリングする experience replay という手法を用いている [23]．ただし，experience replay はメモリを大量に消費するというデメリットが存在する．また，experience replay に蓄えられた経験は過去の学習途中の方策に従って収集されたものである．DQN の学習に使用するデータはどのような方策によって得られたものでも問題ないが，後述する方策勾配法など，現在の方策によっ

て得られたデータを使用しなければならない手法も存在し，そのような場合には experience replay を用いることはできない．DQN のように任意の方策により得られた経験を用いて学習を行える手法を off-policy な手法と呼ぶ．

DQN のように深層学習を用いて強化学習を行う手法を深層強化学習と呼ぶ．

2.1.3 方策勾配法 (Policy Gradient Methods)

Q 学習は最適方策 π における行動の価値 Q^* を学習し，学習した行動の価値を元に最適方策を得ることを目的とした学習だが，方策勾配法 [24] と呼ばれる強化学習の手法では，方策 π を直接更新することで最適方策を得る．パラメータ θ を用いて表される方策 π を更新する際，方策の良さを何らかの指標で表し，その良さの θ に対する微分値を求め，良さが大きくなるように θ を更新することが考えられる．方策勾配法では方策の良さとして，現在の方策 π に従ってエージェントが初期状態から行動した際の期待累積報酬 $\rho(\pi)$ を用いる．すなわち

$$\rho(\pi) = \sum_s d^\pi(s) \sum_a \pi(s,a)\bar{r}(s,a)$$

である．ここで，

$$d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr\{s_t = s|\pi\}$$

である．$Pr\{s_t = s|\pi\}$ は方策 π に従って行動した際に時刻 t で状態 s_t である確率である．すなわち，$d^\pi(s)$ は方策 π に従って行動した際に s が現れる確率を，割引率を考慮して足し合わせたものである．また，$\bar{r}(s,a)$ は状態 s で行動 a を取った際に得られる報酬の期待値である．$\rho(\pi)$ は

$$\rho(\pi) = V^\pi(s_0) = \sum_a \pi(s_0,a)Q^\pi(s_0,a)$$

と表すこともできる．ここで，s_0 は環境の初期状態であり，Q^π は

$$Q^\pi(s_t,a_t) = \mathbb{E}\left[r_t + \sum_{t'=t+1} \gamma^{t'-t} r_{t'}\right]$$

である．ただし，$t' \geq t+1$ において，$s_{t'}$ から $s_{t'+1}$ に遷移する際の行動は方策 π に従って決定する．すなわち，$Q^\pi(s,a)$ は状態 s で行動 a を取った後，方策 π に従って行動した場合の期待累積報酬である．また，$V^\pi(s)$ は状態 s から方策 π に従って行動した際の期待累積報酬である．$Q^\pi(s,a)$ と $V(s)$

については以下の二つの式が成り立つ．

$$V^{\pi}(s_t) = \sum_{a_t} \pi(s_t, a_t) Q^{\pi}(s_t, a_t)$$

$$Q^{\pi}(s_t, a_t) = \sum_{s_{t+1}} T(s_{t+1}|s_t, a_t) \left[r(s_t, a_t, s_{t+1}) + \gamma V^{\pi}(s_{t+1}) \right]$$

なお，$T(s_{t+1}|s_t, a_t)$ は，環境によって定義される状態遷移関数で，状態 s_t で行動 a_t を取った際に s_{t+1} に遷移する確率を表す．また，$r(s_t, a_t, s_{t+1})$ は，状態 s_t で行動 a_t を取り，状態 s_{t+1} に遷移した際に得られる報酬である．

従って，$\rho(\pi)$ をパラメータ θ で微分すると，

$$\frac{\partial \rho}{\partial \theta} = \frac{\partial V^{\pi}(s_0)}{\partial \theta} = \frac{\partial}{\partial \theta} \sum_{a_0} \pi(s_0, a_0) Q^{\pi}(s_0, a_0) \tag{1.1}$$

$$= \sum_{a_0} \left[\frac{\partial \pi(s_0, a_0)}{\partial \theta} Q^{\pi}(s_0, a_0) + \pi(s_0, a_0) \frac{\partial}{\partial \theta} Q^{\pi}(s_0, a_0) \right] \tag{1.2}$$

$$= \sum_{a_0} \left[\frac{\partial \pi(s_0, a_0)}{\partial \theta} Q^{\pi}(s_0, a_0) + \pi(s_0, a_0) \frac{\partial}{\partial \theta} \left[\bar{r}(s_0, a_0) + \sum_{s_1} \gamma T(s_1|s_0, a_0) V^{\pi}(s_1) \right] \right] \tag{1.3}$$

$$= \sum_{a_0} \left[\frac{\partial \pi(s_0, a_0)}{\partial \theta} Q^{\pi}(s_0, a_0) + \pi(s_0, a_0) \frac{\partial}{\partial \theta} \sum_{s_1} \gamma T(s_1|s_0, a_0) V^{\pi}(s_1) \right] \tag{1.4}$$

$$= \sum_{a_0} \left[\frac{\partial \pi(s_0, a_0)}{\partial \theta} Q^{\pi}(s_0, a_0) + \pi(s_0, a_0) \sum_{s_1} \gamma T(s_1|s_0, a_0) \frac{\partial}{\partial \theta} V^{\pi}(s_1) \right] \tag{1.5}$$

$$= \sum_{a_0} \frac{\partial \pi(s_0, a_0)}{\partial \theta} Q^{\pi}(s_0, a_0) + \sum_{a_0} \pi(s_0, a_0) \sum_{s_1} \gamma T(s_1|s_0, a_0) \frac{\partial}{\partial \theta} V^{\pi}(s_1) \tag{1.6}$$

$$= \sum_{a_0} \frac{\partial \pi(s_0, a_0)}{\partial \theta} Q^{\pi}(s_0, a_0) + \gamma \sum_{s_1} \sum_{a_0} \pi(s_0, a_0) T(s_1|s_0, a_0) \frac{\partial}{\partial \theta} V^{\pi}(s_1) \tag{1.7}$$

$$= \sum_{a_0} \frac{\partial \pi(s_0, a_0)}{\partial \theta} Q^{\pi}(s_0, a_0) + \gamma \sum_{s_1} \sum_{a_0} \pi(s_0, a_0) T(s_1|s_0, a_0) \frac{\partial}{\partial \theta} \sum_{a_1} \pi(s_1, a_1) Q^{\pi}(s_1, a_1) \tag{1.8}$$

と表すことができる．ここで，$\sum_{a_0} \pi(s_0, a_0) T(s_1|s_0, a_0)$ は，方策 π に従って行動する場合に $t = 1$

で状態 s_1 をとる確率であることに注意する．式 (1.8) における

$$\frac{\partial}{\partial\theta}\sum_{a_1}\pi(s_1,a_1)Q^\pi(s_1,a_1)$$

も上記と同様の式変形を行うことが可能である．すなわち，

$$\begin{aligned}&\frac{\partial}{\partial\theta}\sum_{a_1}\pi(s_1,a_1)Q^\pi(s_1,a_1)\\ =&\sum_{a_1}\frac{\partial\pi(s_1,a_1)}{\partial\theta}Q^\pi(s_1,a_1)+\gamma\sum_{s_2}\sum_{a_1}\pi(s_1,a_1)T(s_2|s_1,a_1)\frac{\partial}{\partial\theta}\sum_{a_2}\pi(s_2,a_2)Q^\pi(s_2,a_2)\end{aligned} \tag{1.9}$$

と展開することが可能である．これを繰り返すことによって，以下の式が成り立つ．

$$\frac{\partial\rho}{\partial\theta}=\sum_s d^\pi(s)\sum_a\frac{\partial\pi(s,a)}{\partial\theta}Q^\pi(s,a) \tag{1.10}$$

当然ながら，基本的には $Q^\pi(s,a)$ も未知であるため，それも含めて学習する必要がある．

なお，実際に学習を行う際は $A^\pi(s,a)=Q^\pi(s,a)-V^\pi(s)$ を用いて，

$$\sum_s d^\pi(s)\sum_a\frac{\partial\pi(s,a)}{\partial\theta}A^\pi(s,a) \tag{1.11}$$

を用いる場合が多い．これは，以下のようにして式 (1.10) と同一であることを示すことができる．

$$
\begin{aligned}
&\sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} A^\pi(s,a) \\
=&\sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} \left(Q^\pi(s,a) - V^\pi(s)\right) \\
=&\sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a) - \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} V^\pi(s) \\
=&\frac{\partial \rho}{\partial \theta} - \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} V^\pi(s) \\
=&\frac{\partial \rho}{\partial \theta} - \sum_s d^\pi(s) V^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} \\
=&\frac{\partial \rho}{\partial \theta} - \sum_s d^\pi(s) V^\pi(s) \frac{\partial}{\partial \theta} \sum_a \pi(s,a) \\
=&\frac{\partial \rho}{\partial \theta} - \sum_s d^\pi(s) V^\pi(s) \frac{\partial}{\partial \theta} 1 \\
=&\frac{\partial \rho}{\partial \theta}
\end{aligned}
\tag{1.12}
$$

このような A^π をアドバンテージと呼ぶ．$Q^\pi(s,a)$ の代わりにアドバンテージを用いることで，学習時に勾配の分散が減り，学習が安定しやすくなることが知られている．

また，

$$
\frac{\partial \log \pi(s,a)}{\partial \theta} = \frac{1}{\pi(s,a)} \frac{\partial \pi(s,a)}{\partial \theta}
$$

であるので，式 (1.11) は，

$$
\frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} A^\pi(s,a) = \sum_s d^\pi(s) \sum_a \pi(s,a) \frac{\partial \log \pi(s,a)}{\partial \theta} A^\pi(s,a) \tag{1.13}
$$

と変形することができる．すなわち，a について足し合わせる際に，$\pi(s,a)$ で重み付けすることになる．これにより，方策 π に従って行動したエージェントが得た経験から，

$$
L = -\log \pi(s,a) A^\pi(s,a)
$$

の誤差関数で θ を更新することで，ρ に対する確率的勾配降下法として学習が可能であることがわ

かる．なお，この際 $A^{\pi}(s,a)$ の θ に対する勾配を θ の学習時に加算しないように注意が必要である．また，負の符号がついているのは，誤差関数は最小化をすることが一般的であり，今回は期待累積報酬を最大化することが目的であるからである．注意すべき点として，DQN などの Q 学習では学習データとして用いる経験はどのように環境からサンプルしてきても問題なかったが，方策勾配法においては現在の方策 π に従って行動するエージェントの経験を用いて勾配を計算しないと正しく学習ができない．このように現在の方策から得られた経験で学習を行う手法を on-policy な手法と呼ぶ．

2.1.4 探索と利用のジレンマ (exploration-exploitation dilemma)

強化学習において，十分な状態および行動の組に対する経験を収集するためには探索が必要となる．学習途中の方策に従い，報酬を最大化するように行動すると場合によっては局所解に陥ってしまい，大域的にはより多くの報酬を得ることができる方策があるとしてもそこにたどり着くことができなくなってしまう．そのため DQN では先に述べたような ε-greedy 法によりランダムな行動を取ることで探索を行う．しかし ε-greedy 法は場合によっては経験を集めることを阻害する．例えばゲームを最終ステージまで攻略しないと出現しない状態についての経験を得るためには，ゲームオーバーにならずにそのステージまで到達する必要がある．しかし確率 ε でランダムな行動を取ってしまうと，それによりエージェントの状態が悪化し，ゲームオーバーに近づいてしまうことで目的のステージに到達できなくなる可能性がある．このように現在までの学習により得られた方策を活用してより先の状態を経験するための活用（exploitation）と，現在の方策では見逃されている状態を経験するための探索（exploration）の両立は難しく，強化学習の課題となっている．また，方策勾配法などの on-policy な手法においては現在の方策に従って得られた経験のみから学習が可能であるため，探索を行うことが困難である．

近年，好奇心ベースの手法 [25, 26] により，未知の状態に対して報酬を与えることで探索を行う手法が提案され，いくつかのゲームにおいて既存手法を上回るスコアを獲得している．

2.2 強化学習の分散アーキテクチャ

深層強化学習においても方策勾配法を利用することで，より効率よく学習を進めることが考えられる．方策勾配法は on-policy な手法なので，DQN のように experience replay を用いることができない．experience replay を用いずにサンプル間の相関を減らし，また学習の効率を高めるために，環境を複数並列に実行してサンプルを取得する手法が提案されている．

2.2.1 Asynchronous Advantage Actor-Critic (A3C)

2016 年，Mnih らにより方策勾配法を用いた深層強化学習の手法として A3C (asynchronous advantage actor-critic) が提案された [8].

A3C では，方策を表す方策関数 $\pi(a_t|s_t;\theta)$ と $V^\pi(s_t)$ を近似するための価値関数 $V(s_t;\theta_v)$ のニューラルネットワークをそれぞれ学習する．π は行動を決定するので Actor と呼ばれ，V を用いて現在の方策の善し悪しを判断するため，V は Critic と呼ばれる．

CPU のコア一つに Actor と Critic を一組ずつ対応させることで，並列計算による高速な学習を可能とする．また，各 CPU コア上の Actor, Critic が別の環境上で実行され，並行してパラメータの更新を行うため，データの相関が薄れ，experience replay を用いなくても安定した学習が可能である．各コアで勾配を計算した後，それぞれのコアで計算された勾配を足し合わせた上でパラメータの更新を行う．

学習における誤差の計算法としては，以下のアドバンテージを用いることが考えられる．

$$A^\pi(s_t,a_t) = Q^\pi(s_t,a_t) - V^\pi(s_t,a_t) = \sum_{s_{t+1}} T(s_{t+1}|s_t,a_t)\left[r(s_t,a_t,s_{t+1}) + \gamma V^\pi(s_{t+1})\right] - V^\pi(s_t)$$

エージェントが s_t で方策 π に従って a_t を選択し，s_{t+1} に遷移して報酬 r_t を得たとすると，

$$A^\pi(s_t,a_t) = r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$$

を学習データとして用いることができる．V の学習では，

$$L_V = \left[r_t + V(s_{t+1}) - V(s_t;\theta)\right]^2$$

を用いる．この際 $V(s_{t+1})$ も θ に依存するが，DQN の target network と同様に，こちらの勾配は計算しない．π については，方策勾配法により，

$$L_\pi = -\log\pi(s_t,a_t)A(s_t,a_t)$$

を用いて学習を行う．A3C ではこの際，数ステップの連続した経験を用いることでより正確なアドバンテージを推定することを提案している．すなわち，$t=t_s$ から $t=t_s+N$ までの間方策 π に従っ

て行動した経験がある場合，

$$A(s_{t_s}, a_{t_s}) = \sum_{t=t_s}^{t_s+N-1} \gamma^{t-t_s} r_t + \gamma^N V(s_{t_s+N}) - V(s_{t_s})$$

のようにアドバンテージを計算できる．

A3C は on-policy な手法であるため，ε-greedy などの探索を行うことができない．局所解に陥ることを防ぐために，A3C では以下のようなエントロピー誤差に小さな係数をかけて誤差関数に加えることが提案されている．

$$L_{entropy} = \sum_a \pi(s, a) \log \pi(s, a)$$

論文中の実験では，16 コアの CPU を用いて学習を行い，単一の GPU で学習した DQN と比較して学習効率が大きく上回っていることを示している．

2.2.2 Hybrid CPU/GPU A3C (GA3C)

A3C は各 CPU で勾配を計算するため，GPU での並列計算が行えず，計算時間が比較的長くなりがちであるという欠点があった．GPU での並列計算は，一度に複数のデータを計算することで高い効率を実現しており，環境が非同期に実行され，ニューラルネットワークの計算を必要とするタイミングが環境毎に異なる A3C とは相性が悪かった．ニューラルネットワークの計算が必要になるのは，エージェントの行動時に，行動を決定するための $\pi(s_t, a_t)$ およびアドバンテージ計算用の $V(s_t)$ を求める際と，実際に行動を行って得られた経験から π と V を学習する際である．Batched A2C [27] ではそれらのタイミングで並列で動作する各環境からのデータがそろうのを待ち，GPU で並列計算を実行するが，最も計算が遅れた環境を待つ必要があるため，実行効率が低いという欠点がある．

そこで，データをキューに積むことで実行効率を上げつつ A3C の計算を GPU 上で行う Hybrid CPU/GPU A3C (GA3C) という手法が Babaeizadeh らによって提案された [28]．GA3C では CPU のコア数よりも多くの環境を並列で実行し，ニューラルネットワークの実行に必要なデータをキューに積み，キューからバッチを作成して順次 GPU に送ることで，非常に高い効率で学習を行う事ができる．

通常の A3C や Batched A2C [27] とは異なり，行動を行った後，その行動によって得られた学習データが学習されるまでに時間差があるため，その間に他の環境から得られた学習データによる学習が行われる場合があり，行動時の方策と学習時の方策にずれが生じる可能性がある．方策勾配法は on-policy な手法であるため，方策のずれは学習に悪影響を与える．GA3C では，以下のように方

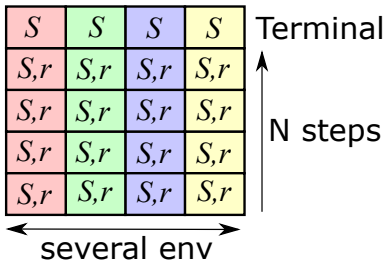

図 2.1: IMPALA における学習データの構成

策関数の値に微少量 ε を加えることを提案している．

$$L_\pi = -\log\left[\pi(s_t, a_t) + \varepsilon\right] A(s_t, a_t)$$

これにより学習が安定化し，方策のずれによる影響を小さく抑えることができる．

2.2.3 Importance Weighted Actor-Learner Architecture (IMPALA)

GA3C では行動時と学習時の方策のずれに対して単に微少量 ε を方策関数の値に加えることで対応したが，より正確に誤差を求めるために，Importance Sampling を用いる Importance Weighted Actor-Learner Architecture (IMPALA) が Espeholt らによって提案された [19]．IMPALA では GA3C と同様に，キューに積まれたデータを順次 GPU に送り計算する方式をとるが，方策のずれに対応するために Importance Sampling を行っている．それに加え，アドバンテージの計算を学習時まで遅延する手法をとっている．GA3C においては，行動を決定する際に現在の状態 s_t を GPU に送るが，その際 s_t に対する方策だけでなく，価値関数 $V(s_t)$ の値も同時に取得する．ここで取得した価値関数の値を用いてアドバンテージの計算を行っていたが，価値関数も学習時までに変化する可能性があるため，学習の正確性を損なっていた．IMPALA においては，図 2.1 に示すように学習データに次状態の観測を付与し，実際に学習する際に価値関数に通すことでアドバンテージの計算を行っている．なお，図 2.1 における S は状態，r は報酬を表す．図では省略したが，このほかに実際に選択した行動と，その行動を行動を取った際の $\pi(s, a)$ の値も必要となる．従来の A3C や GA3C では各環境においてアドバンテージを計算していたが，IMPALA では複数環境から得られた学習データをバッチ化して GPU に送り，終端状態の価値関数を計算してからアドバンテージを求める必要がある．アドバンテージの計算は n-step 先から逆順に行う必要があるため，IMPALA における学習データのバッチには時系列情報を含める必要がある．その上で，行動時に方策において実際の行動 a_t を選択する

確率 $\mu(a_t|s_t)$ と，学習時の方策において a_t を選択する確率 $\pi(a_t|s_t)$ の比

$$\rho_t = \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}$$

を用いて価値関数の値に補正をかけることで，行動時と学習時の方策関数のずれを適切に計算して学習を行うことが可能となる．

第3章　モデルベース強化学習

3.1　環境モデル

環境モデルとはすなわち状態遷移関数 $T(s_{t+1}|s_t, a_t)$ と報酬関数 $r(s_t, a_t, s_{t+1})$ のことであり，全ての状態および行動に対してこれらの値が計算可能なとき，その環境を既知であるという．既知な環境においてはエージェントは先読みを行うことができる．すなわち現在の状態 s_t から仮想的に状態遷移を繰り返し，将来の状態および報酬を計算することで，最終的な方策を得るという手法をとることができる．従来のボードゲームなどにおける AI はこのような探索によるものが多かったが [1]，強化学習の研究においては環境は未知である設定が一般的であるため，エージェントは仮想的な状態遷移に基づいた探索を行うことができない．これを可能とするために，エージェント自身の経験から環境モデルを学習し，学習した環境モデルを用いて探索を行う手法が提案されており [11]，これらの手法をモデルベース強化学習と呼ぶ．これに対し，従来の Q 学習などのように環境のモデルを陽には用いずに方策を学習する手法をモデルフリー強化学習と呼ぶ．近年のモデルベース強化学習に関する研究では，環境モデルをニューラルネットワークで近似し，エージェントの経験により実際に得られた次状態や報酬を教師データとしてネットワークの学習を行うものが多い．

3.2　モデルベース強化学習の課題

強化学習における目的は累積報酬の最大化である．モデルフリー強化学習においても累積報酬の最大化を目的に学習が行われており，学習が完全に行われれば最適な方策を得ることが可能である．しかしビデオゲームなどの複雑な環境において完全な学習を行うことは困難である．特に現在の状態から離れた将来の報酬について正しく学習することは非常に困難である．その結果，モデルフリー強化学習は将来を見据えた長期的な計画に基づく方策を学ぶことを苦手とする傾向にある．モデルベース強化学習は環境モデルを用いた探索を行うことでこの欠点をある程度克服することができ，また学習データに対するサンプル効率が良いことが知られている [12]．

しかし深層強化学習を用いたモデルベース強化学習にはいくつかの課題が存在する．第一に，不完全な環境モデルを用いた先読みによる性能の低下である．複雑な環境において環境モデルの学習を完全に行うことは困難である．そのため学習した環境モデルによる次状態予測にはある程度の誤差が存在する．この状態で先読みを繰り返すと，誤差の蓄積により予測された状態から情報を得ることが困難になる．実際に学習した環境モデルで単純なモンテカルロ木探索などを行うとモデルフリーな手法と比較して性能が悪化することが確認されている [15]．この問題については，予測により得られた状態の軌跡全てをニューラルネットワークに入力し，環境モデルの信頼度も含めて適切な情報抽出を行えるように学習する手法が提案されている [15, 16]．

また，深層強化学習においては観測としてゲーム画面などの高次元のデータが用いられることが多い．この観測データを用いて環境モデルを定義すると，行動と高次元のデータから高次元のデータを出力するモデルとなる．これをニューラルネットワークで表現すると，環境モデルの学習の際や方策の学習時，テスト時において多くの計算リソースを必要とする．ロボットの制御などは短い時間で行動を選択する必要がある場合が多く，高次元の環境モデルによる計算時間の増大は現実の複雑なタスクに適用する際に問題となることが考えられる．この問題を解決するため，観測を低次元の中間表現に抽象化し，その中間表現の空間で状態遷移を学習する手法が複数提案されている．

3.3 Imagination-Augmented Agents (I2A)

不完全な環境モデルから予測された状態を適切に扱うことができるモデルベース強化学習のアーキテクチャとして，Imagination-Augmented Agents (I2A) が Weber らによって提案された [15]．I2A では，観測画像と行動から次の観測画像を直接予測する環境モデルを事前学習して方策の決定に用いる．観測として与えられる画像は一般に大きな次元を持ち，予測は高計算コストかつ複雑になる．そのため，モデルの正確性は担保できないが，予測された画像を畳み込みニューラルネットワークに通して特徴抽出した後，Long Short-Term Memory（LSTM） [29] を用いたエンコーダによって時系列の逆順にエンコードすることで，モデルの不正確さを含めてネットワークに学習させ，最適な方策を得るという手法をとっている．具体的には，まず図 3.1 に示すように環境モデルによる予測を n ステップ先まで繰り返す．予測途中の行動の決定にはロールアウトポリシー $\hat{\pi}$ を用いる．$\hat{\pi}$ は観測を入力として取り，エージェントが取るであろう行動を予測して出力するニューラルネットワークである．ロールアウトポリシーにより予測された行動と現在の観測を環境モデルに与え，次の観測を予測する．これを n ステップ繰り返した後，予測された観測及び報酬を，図 3.2 に示すように時系列の逆順に LSTM を用いたエンコーダに入力していく．学習された環境モデルは基本的には確

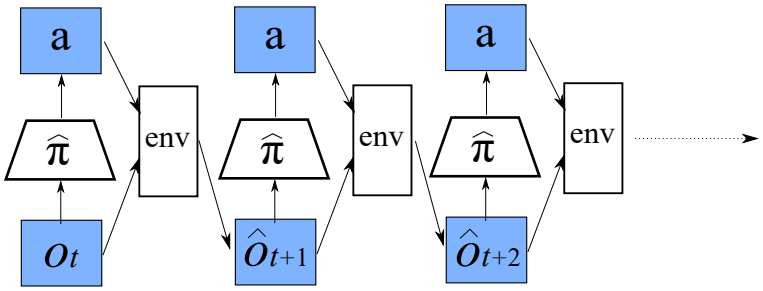

図 3.1: I2A における次状態予測

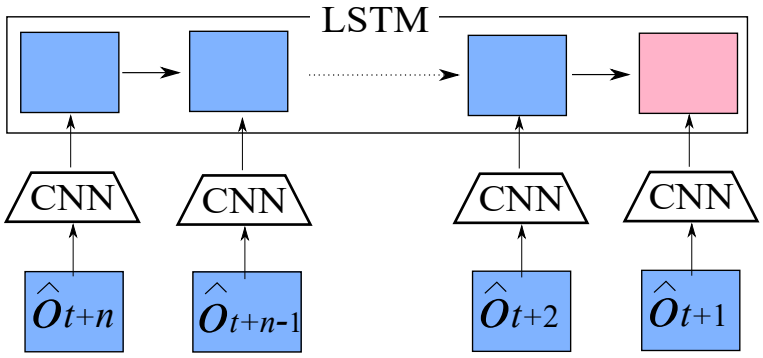

図 3.2: I2A における予測された次状態からの特徴抽出

定的に振る舞うが，ロールアウトポリシー $\hat{\pi}$ は一般的には行動の確率分布を出力するため，そこから行動 a をサンプルする際に確率的な思考を伴う．よって，予測によって現れる次状態の軌跡は複数回の試行を行うと異なるものになるため，I2A では複数の試行（ロールアウト）を行う．複数回の試行によってそれぞれエンコードされた出力全てと，通常のモデルフリーな畳み込みニューラルネットワークによって出力された隠れ層とを結合し，価値関数及び方策関数の計算に用いる．$\hat{\pi}$ は最終的な方策関数 π に近い値を出力することで，エージェントが将来実際に取るであろう行動と同様の行動で予測を行えるため，予測が有用なものになりやすい．そのため，図 3.3 に示すように学習途中に実際の π の値を教師として $\hat{\pi}$ を蒸留することで，より精度の高い予測を行うことができる．

I2A は倉庫番とミニパックマンのゲームにおいて，モデルフリーの A3C よりも高いスコアを記録したことが報告されている．またミニパックマンにおいては，学習した環境モデルを変更することなく複数種類のタスクにて高いスコアを獲得しており，モデルベース強化学習の利点の一つであるタスクに関する汎用性が確認された．

I2A の環境モデルは観測である画像と行動を入力とし，直接次の観測に当たる画像を出力しているため，計算コストが大きいという欠点がある．

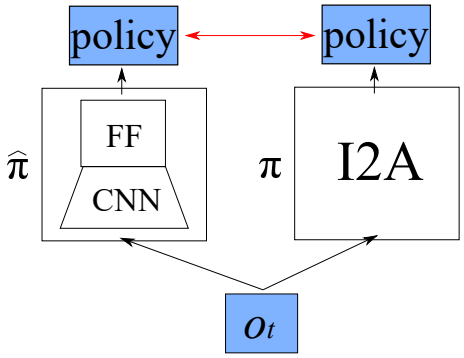

図 3.3: I2A における方策の蒸留

3.4　抽象化された状態表現を用いた強化学習

高次元な観測をそのまま環境モデルの入出力として用いると，計算時間の増加を招く．そこで図 4.1 に示すように，高次元の観測を低次元の中間表現に抽象化して，その中間表現の空間で状態遷移を学習する手法が提案されている．画像が入力として用いられる場合，低次元のベクトル表現への変換は畳み込みニューラルネットワーク（Convolution Neural Network, CNN）が用いられることが多いが，CNN を学習するための誤差関数は手法により様々な種類がある．既存手法の多くは中間表現の事前学習が必要であるなどの課題を抱えている．

3.4.1　World Models

Ha らは 2018 年，World Models と呼ばれるモデルベース強化学習の手法を提案した [17]．環境の遷移モデルに画像を直接用いるのではなく，事前に学習したニューラルネットワークにより次元数を削減した中間表現を用いる．World Model は環境から観測として得られた画像を Variational Autoencoder [30] でエンコーディングする．Variational Autoencoder では，画像を一度低次元の表現に変換した後，元の画像を復元し，元の画像を教師データとして学習を行う．すなわち，元画像を復元可能な中間表現を得るための学習手法である．これはタスクによらない特徴量であるため，モデルベース強化学習の，報酬が変化しても環境モデルを再利用できる利点を活かすことができる．その反面，タスク達成のために必要な情報の重要度が反映されにくい可能性が懸念される．

World Model では状態遷移モデルに MDN-RNN [31] を用いており，従来の手法では考慮できなかった確率的な状態遷移を扱っている．また，環境が確率的に遷移することにより，不正確な環境モデルを用いた先読みによる悪影響を軽減する効果があるとしている．

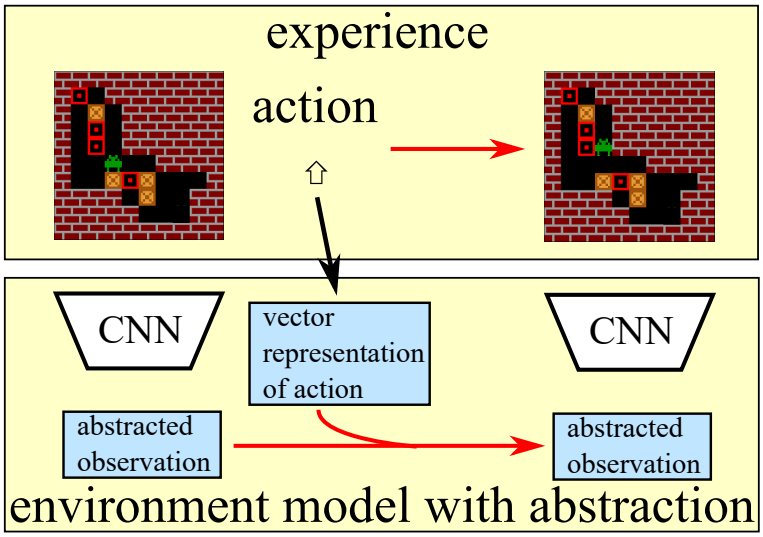

図 4.1: 抽象化された状態表現を用いた環境モデル

World Model ではベクトル表現及び状態遷移の学習のため，事前にランダムエージェントにより集めた観測を利用する．そのため，ゲームを攻略しないと現れない状態に対応することが困難である．論文中では学習したエージェントの行動をもとに再度 Variational Autoencoder 及び状態遷移モデルの学習を行うことを繰り返す手法も提案されているが，学習コストが大きくなるという欠点がある．

3.4.2 State-space models (SSM)

Buesing らが 2018 年に発表した State-space models (SSM) では，環境の状態遷移を含めて観測画像を復元できるような中間表現の学習を行う [18]．すなわち，図 4.2 に示したように，エージェントの経験から観測 o_t と o_{t+1} が得られた場合，o_t を変換した低次元の h_t から何らかの操作により h_{t+1} を求め，それを元に o'_{t+1} を計算する．実際の o_{t+1} を教師データとして，h_t から h_{t+1} への変換部，s_t から h_t を得る CNN および h_{t+1} から o'_{t+1} への変換を行う Decoder を学習する．状態遷移を含めて学習しているため，World Model で用いられた Variational Autoencoder よりも環境の特徴を効率よく表現した中間表現が得られることが期待される．しかし，誤差計算の際に高次元のデータ同士を比較するため，環境モデルの学習中常に高コストな計算を行う必要がある．また，SSM でも環境モデルおよび中間表現を事前学習によって得るため，World Model と同様に，方策学習時と表現学習時の観測分布が異なる問題が発生しうる．

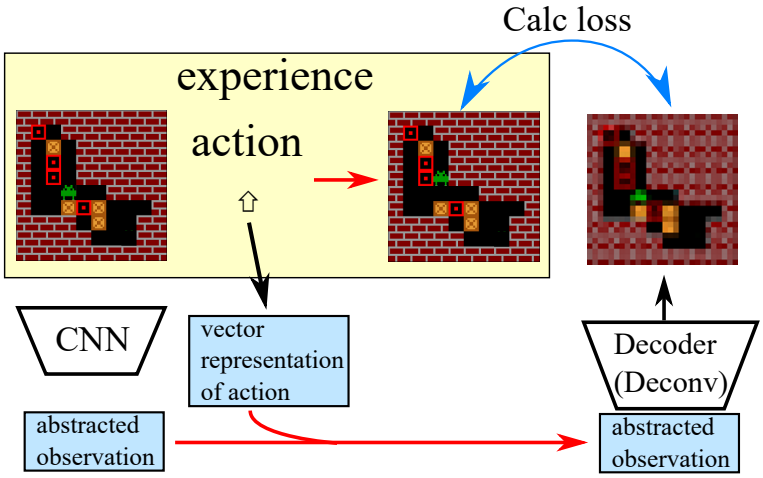

図 4.2: SSM の概要

3.4.3 Reinforcement Learning with Hidden Layer Predictor (RL-HLP)

亀甲らは 2017 年，A3C の隠れ層の状態遷移を学習することで先の状態を用いた方策決定を可能とする，Reinforcement Learning with Hidden Layer Predictor (RL-HLP) を提案した [16]．まず，事前に学習した A3C のエージェントから隠れ層を計算する畳み込みニューラルネットワーク $h^t_{A3C} = \mathrm{CNN}(s_t;\theta_{cnn})$ 及び隠れ層から方策を決定するフィードフォワードニューラルネットワーク $\pi^t = \mathrm{Softmax}(\mathrm{FF}(h^t_{A3C};\theta_\pi))$ を得る．これらを用いて，隠れ層 h^t_{A3C} 及び行動 a から次の隠れ層 $h^{t+1}_{pred,a}$ を計算するネットワークを学習する．このネットワークは以下の式に示した通り，h^t_{A3C} を LSTM の隠れ層として，状態 a を LSTM の各ステップの入力に用いて実現される．$\mathrm{Embed}(a)$ は a のベクトル表現であり，θ_{rnn} と同時に学習される．

$$h^{t+1}_{pred,a_i} = \mathrm{LSTM}(h^t_{A3C}, \mathrm{Embed}(a); \theta_{rnn})$$

World Model や SSM では実際に経験で得られた観測を復元するような誤差計算を行っていたため，高次元のデータを比較する必要があったが，RL-HLP は事前の強化学習によって得られた CNN を用いて中間表現を得た上で，あくまでも次状態の中間表現を予測するように誤差計算を行うため，比較的計算量を抑えやすい．この LSTM を用いて与えられた観測から各行動を取った場合の次状態（隠れ層）を計算し，そこから n ステップ先までの隠れ層を予測して方策の決定に用いる．予測された隠れ層から次の隠れ層に遷移する際の行動決定には事前に学習された $\mathrm{FF}(h^t_{A3C};\theta_\pi)$ が用いられる．予測された隠れ層をすべて結合し，全結合ネットワークに通した後，さらに元の隠れ層 h^t_{A3C} と結合

して全結合ネットワークにより価値関数と方策関数の値が決定される．途中の状態を含めてニューラルネットワークに渡すことで，環境モデルの信頼度なども含めた特徴抽出をする点は I2A [15] と類似している．RL-HLP は，Atari 2600 のいくつかのゲームにおいて，従来の A3C を上回る性能を記録したことが報告されている．

ただし，RL-HLP は途中の状態（隠れ層）を全て結合して全結合ネットワークの入力としているため，予測するステップ数の増加に比例して学習パラメータが増加する．また将来的に，動的に先読みを行うステップ数を変更するなどの変更を加えにくい．別のモデルにおいて，動的に先読みの方法を変更する研究が既に存在する [32] ため，予測ステップ数が固定であるという制約が問題になることは十分考えられる．

また，予測に用いる CNN($s_t;\theta_{cnn}$) 及び FF($h^t_{A3C};\theta_\pi$) は，RL-HLP の学習中は固定されているため，事前学習の精度によって性能の上限が決まってしまうという問題がある．

3.4.4 Combined Reinforcement via Abstract Representations (CRAR)

観測の中間表現の獲得と強化学習を同時に行う手法としては，François-Lavet らによって提案された Combined Reinforcement via Abstract Representations (CRAR) が存在する．CRAR では DQN による強化学習と同時に，DQN の畳み込みニューラルネットワーク部分により得られた中間表現を用い，その中間表現の空間で状態遷移を学習する．しかし CRAR では，強化学習の学習においてはあくまでもモデルフリーかつ off-policy な DQN を用いており，環境モデルの学習は特徴表現獲得の補助として利用されている．環境モデルの学習時の誤差を強化学習の誤差に加えて畳み込みニューラルネットワークの学習に利用することで，より方策決定に適した中間表現が得られるとしている．テスト時には環境モデルを用いたロールアウトを行うこともできるとしているが，その際は単に Q 値を合計しており，環境モデルが不正確な場合は大きな性能向上は見込めない．

3.4.5 抽象化された状態表現の獲得

RL-HLP や CRAR では，強化学習のエージェントにより得られた畳み込み層を利用して中間表現を得ている．World Model では Variational Autoencoder を用いることで，ゲーム画面における画像としての特徴表現を利用している．Universal Planning Networks (UPN) [33] では，エキスパートの動作を模倣する学習をする過程で，タスク達成に適した中間表現の獲得を行っている．Contrastive Predictive Coding [34] では，数ステップ先の予測とネガティブサンプリングを組み合わせることに

より次状態予測に有用な表現を得ている．Burda らは，好奇心ベースの学習に用いる次状態予測において，Variational Autoencoder や Inverse Dynamics に基づくものなど，いくつかの中間表現を比較している [35].

強化学習の目的は，環境の中で累積報酬を最大化すること，すなわちタスクを効率的に達成することである．Variational Autoencoder などのタスクに依存しない中間表現は，報酬の変化に柔軟に対応できる半面，タスク達成という観点では効率の悪い表現になる恐れがある．UPN はタスク達成に適した表現をエキスパートの動作を模倣することで学習するが，教師となるデータを必要とする．

また，強化学習ではタスクをこなしていくことにより，ゲームのステージが進むなどして，観測データの傾向が大きく変わることがある．すなわち，性能の低い方策と性能の高い方策では，観測データの分布が異なるということである．そのため，上手くタスクをこなすことができない状態で得られるデータは不十分である場合がある．したがって中間表現を事前学習する手法に関しては，事前学習の時点で得られたデータの傾向に性能が依存してしまう危険がある．World Model では，学習とデータ収集を繰り返し行うことでこの問題に対応する手法が提案されているが，学習効率が落ちる上に，繰り返しの周期を調整する必要が生じてしまう．

第4章　タスク依存の中間表現を用いたアーキテクチャの提案

本論文で提案するモデルベース強化学習のアーキテクチャでは，画像などの高次元の観測をタスク達成に適した低次元の中間表現に変換し，その中間表現の空間で次状態を予測する．また，予測された次状態を Gated Recurrent Unit (GRU) [36] を用いてエンコードして最終的な方策を決定する．環境モデルを用いた予測により得られる特徴量に結合されたモデルフリーな特徴量を中間表現として利用することで，これらの学習を同時に効率よく行うことができる．

4.1　アーキテクチャの概要

アーキテクチャの概要を図 1.1 及びアルゴリズム 1 に示す．図 1.1 における白い長方形には学習対象のパラメータが存在する．赤色の長方形は環境から得られたある時刻における観測であり，青色の長方形はその観測に対して行う計算の途中で現れるベクトルである．エージェントは始めに畳み込みニューラルネットワークを用いて観測された状態 o を中間表現 h へと変換する．その後，全ての取りうる行動 a_i に対して以下の操作を行う．まず，a_i のベクトル表現を GRU の入力とし，GRU の隠れ状態を h として次状態の中間表現を予測する．このとき次状態予測に用いる GRU を Predictor とする．Predictor により予測された次状態 h_{pred} を全結合ニューラルネットワークを用いたロールアウトポリシー $\hat{\pi}$ に入力し，次の行動を予測する．予測された行動を Predictor に入力し，更に次の状態の予測を行う．これを N ステップ繰り返した後，予測された状態を時系列の逆順に GRU に入力し，エンコードされた特徴量を得る．全ての a_i について特徴量を得た後，それら及び最初に求めた h を結合し，それを全結合ニューラルネットワークに入力することで，方策関数及び価値関数の値を得る．

学習初期においては，Predictor が予測する次状態は不正確なものであるが，方策決定においては中間表現 h を予測により得られた特徴量に結合して用いているため，h を利用してよりよい方策が学習されることが期待される．すなわち h が方策決定に有用な表現となるように学習が進む．この

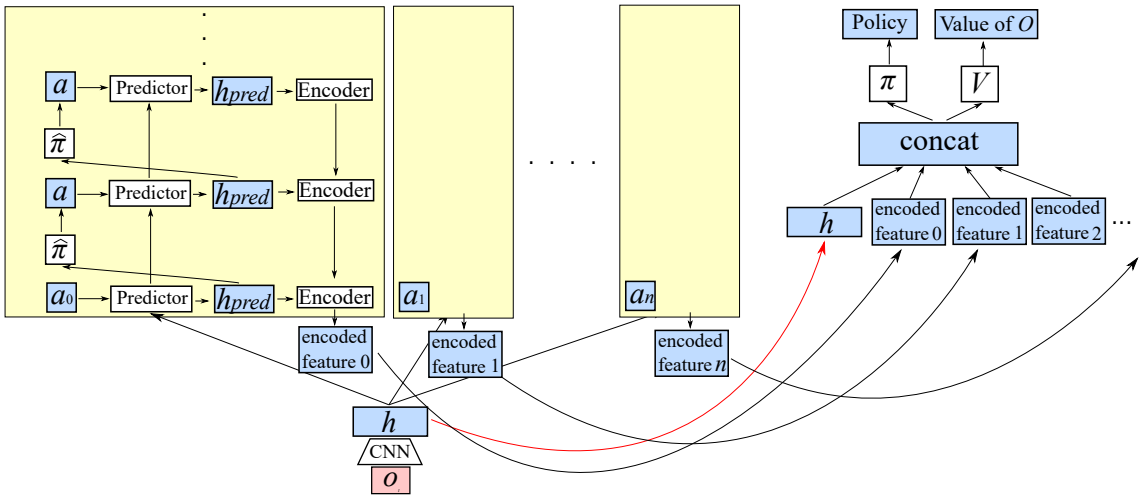

図 1.1: 本稿で提案するアーキテクチャの概要

点に関してはモデルフリーな手法と同様である．学習が進むに連れ，h がタスク達成に適した中間表現になり，また Predictor の精度も向上していくことが期待される．このように学習が進んだ後は，Predictor の出力結果を利用することで，h のみに頼る場合よりも更によい方策を学習することが可能となる．

4.1.1　環境モデルの学習

Predictor 及び行動のベクトル表現の学習は，学習中にエージェントが実際に取った行動及び観測された状態を教師として行う．予測誤差としては，UPN [33] に倣い，実際の中間表現と予測された中間表現の Huber Loss を用いた．本論文では学習に IMPALA [19] を用いたが，IMPALA の学習データには既に時系列情報が含まれているため，状態予測の学習のために追加の情報は不要である．すなわち，学習データ中には状態 s_t で行動 a_t を取った際の次状態 s_{t+1} の情報が常に含まれている．終端状態については行動および次状態の情報は含まれないが，この状態は次の学習データに学習対象として含まれる．また，学習における教師データとなる隠れ状態 h は価値関数及び方策関数の値を計算する際に必要となるため，それらと同時に学習することで重複する計算を省き，効率的な学習を行うことができる．状態予測の学習誤差を計算する際は，教師データとなる h は定数として扱い，h に対する勾配は計算しない．これは，h はあくまでタスク達成に有用な表現となることを期待しているためであり，予測誤差により h を変更してしまうと，予測が容易になるような表現が学習されてしまう恐れがあるためである．逆に，価値関数及び方策関数による強化学習の誤差計算時に

アルゴリズム 1 policy and value function

o is an observation from an environment
$h^0 \leftarrow \mathrm{CNN}(s)$
for $a^1 = 1, 2, \cdots, n$ **do**
  $h^1_{pred} \leftarrow \mathrm{Predictor}(h^0, a^1)$
  for $t = 2, 3, \cdots, M$ **do**　▷ predict for M steps
    $a^t_{pred} \leftarrow \mathrm{Sample}(\hat{\pi}(h^{t-1}_{pred}))$
    $h^t_{pred} \leftarrow \mathrm{Predictor}(h^{t-1}_{pred}, a_{pred})$
  end for
  $encoded_{a^1} \leftarrow \mathrm{Encoder}(h^M_{pred}, h^{M-1}_{pred}, \cdots, h^1_{pred})$
end for
$f \leftarrow \mathrm{concat}(h^0, encoded_0, encoded_1, \cdots, encoded_n)$
$\pi \leftarrow \mathrm{softmax}(W_\pi f + b_\pi)$
$V \leftarrow W_V f + b_V$

は h_{pred} を定数として扱い，Predictor 及び行動のベクトル表現には影響を与えないようにした．

4.1.2　蒸留によるロールアウトポリシーの学習

ロールアウトポリシー $\hat{\pi}$ に関しては，I2A [15] と同様に，学習データにおける h から得られた π と $\hat{\pi}$ が等しくなるように以下に示す蒸留誤差 $L_{dist}(\pi, \hat{\pi})$ を与えて学習を行った．この際にも h は定数として扱った．

$$L_{dist}(\pi, \hat{\pi}) = \sum_{a=1}^{n} \pi(a|h) \log \hat{\pi}(a|h)$$

4.1.3　モデルの学習

本提案手法を用いたモデルの学習における誤差関数は以下のように表すことができる．

$$L = L_{RL} + L_{pred} + L_{dist}\,,$$

ここで，L_{RL} は強化学習に関する誤差を表し，通常の IMPALA と同様に計算された価値関数及び方策関数の誤差，およびエントロピー誤差が含まれる．

この誤差関数の勾配は以下のように計算される．

$$\frac{\partial}{\partial \theta} L = \frac{\partial}{\partial \theta_{RL}} L_{RL} + \frac{\partial}{\partial \theta_{pred}} L_{pred} + \frac{\partial}{\partial \theta_{\hat{\pi}}} L_{dist}\,.$$

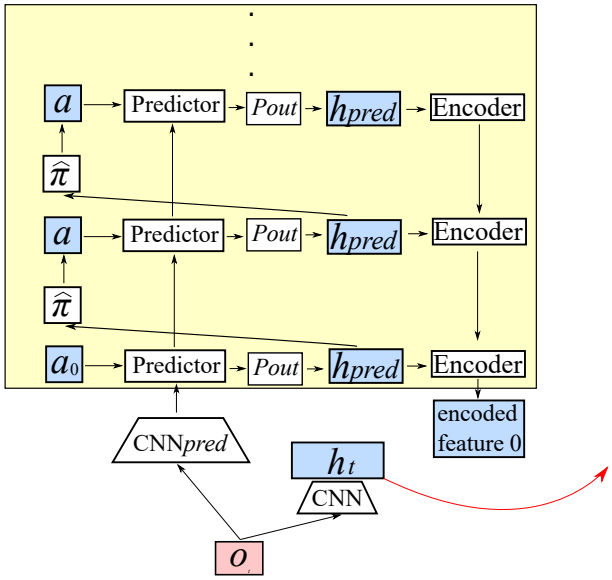

図 1.2: Separate hidden model における Predictor

上記式において， θ_{RL} は CNN および Encoder，π と V のパラメタを表す．θ_{pred} は Predictor および行動のベクトル表現を表す．

4.1.4　Predictor の隠れ層と中間表現を分離したモデル

中間表現である h は価値関数及び方策関数から計算された誤差のみで学習されており，次状態予測により得られた誤差は CNN に伝搬しないように設計されている．よって，環境によっては次状態の h を予測するために h の情報のみでは不十分になることが考えられる．すなわち，h は現在の観測に対する価値関数や方策関数を出力するのに十分な情報を含んでいるが，次状態を予測するための情報は欠落しているといった学習が起こりうる．そのため，より正確な次状態予測を可能とするために，図 1.2 に示すような予測器を用いることが考えられる．本稿では以後この予測器を用いるモデルを Separate hidden model と呼ぶ．Separate hidden model では，Predictor の隠れ状態と h を同一視せず，Predictor の出力の後に出力層 P_{out} を通すことで h_{pred} を得る．また，Predictor の隠れ状態の初期値に h をそのまま用いることができないため，観測 o から Predictor の初期値を得るための畳み込みニューラルネットワークが必要になる．現在の観測 o に対応する h は予測には用いられないが，通常のモデルと同様に予測により得られた特徴量と結合して方策決定に用いられる．こうして学習された h が Predictor 学習時の教師データとして用いられる点は通常のモデルと同様で

アルゴリズム 2 policy and value function (Separate hidden)

o is an observation from an environment
$h^0 \leftarrow \mathrm{CNN}(s)$
$x^0 \leftarrow \mathrm{CNN}_{pred}(s)$
for $a^1 = 1, 2, \cdots, n$ **do**
  $x^1 \leftarrow \mathrm{Predictor}(x^0, a^1)$
  $h^1_{pred} \leftarrow W_{P_{out}} x^1 + b_{P_{out}}$
  for $t = 2, 3, \cdots, M$ **do**　▷ predict for M steps
    $a^t_{pred} \leftarrow \mathrm{Sample}(\hat{\pi}(h^{t-1}_{pred}))$
    $x^t \leftarrow \mathrm{Predictor}(x^{t-1}, a_{pred})$
    $h^t_{pred} \leftarrow W_{P_{out}} x^t + b_{P_{out}}$
  end for
  $encoded_{a^1} \leftarrow \mathrm{Encoder}(h^M_{pred}, h^{M-1}_{pred}, \cdots, h^1_{pred})$
end for
$f \leftarrow \mathrm{concat}(h, encoded_0, encoded_1, \cdots, encoded_n)$
$\pi \leftarrow \mathrm{softmax}(W_\pi f + b_\pi)$
$V \leftarrow W_V f + b_V$

ある．Separate hidden model のアルゴリズムをアルゴリズム 2 に示した．

4.1.4.1　畳み込み層の共有

Separate hidden model では，畳み込みニューラルネットワークの計算を二度行う必要があり，計算時間の面で問題になることが懸念される．また畳み込み層で特徴を抽出する面において別々に学習を行う必要があり，サンプル効率も低下する可能性がある．これらの問題を解決するために，図 1.3 に示すように畳み込み層を共有するモデルが考えられる．h および Predictor の隠れ層を得るための CNN では，何層かの畳み込み層の後に一層の全結合層を用いて所望の次元のベクトルを得ることが一般的であると考えられる．そのようなネットワークの畳み込み層の部分のみを共有し，最後の全結合層を別々に用意することで，計算時間を削減するとともに学習のサンプル効率が上昇することが期待される．

4.1.4.2　報酬を予測する機構の追加

h と Predictor の隠れ層を分けることにより，次状態の h のみならず図 1.4 に示すような機構で報酬を予測することが考えられる．Separate hidden model では Predictor の出力隠れ層に対して全結

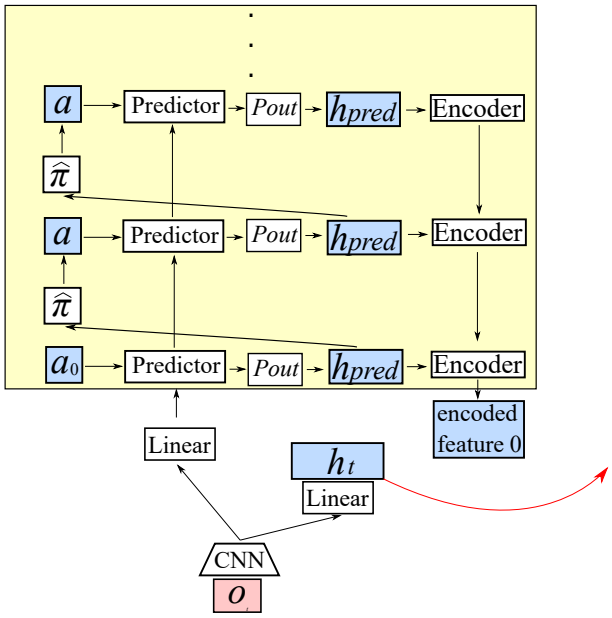

図 1.3: Separate hidden model における畳み込み層の共有

合層 P_{out} を適用することで次状態の h を予測していたが，この際別の R_{out} を適用することで次状態に遷移した際の報酬を予測する．予測した報酬 R_{pred} は h_{pred} と結合して Encoder に入力する．

4.1.4.3　Predictor の隠れ層を Encoder に入力するモデル

Separate hidden model においても Predictor の学習は次状態の h を教師として行われるが，Encoder に入力する際においては h_{pred} ではなく図 1.5 のように Predictor の隠れ層をそのまま用いることも考えられる．Predictor の隠れ層は次状態だけでなくより先の h_{pred} を出力するために十分な情報を含んでいると考えられるため，それをエンコードすることでより有用な特徴が得られる可能性がある．

4.1.5　既存手法との比較

提案手法は，I2A [15] と同様に，予測された状態をリカレントニューラルネットワークを用いてエンコードしているため，不正確な環境モデルから得られた予測データの適切な扱い方を含めてネットワークに学習させることができる．また，画像を直接入力することなく，画像から抽出された特徴ベクトルを予測するため，計算コストを抑えることができ，広範囲への応用が可能である．個々の工夫に関しては既存手法に類似のコンポーネントが存在するが，提案するアーキテクチャは，不正確

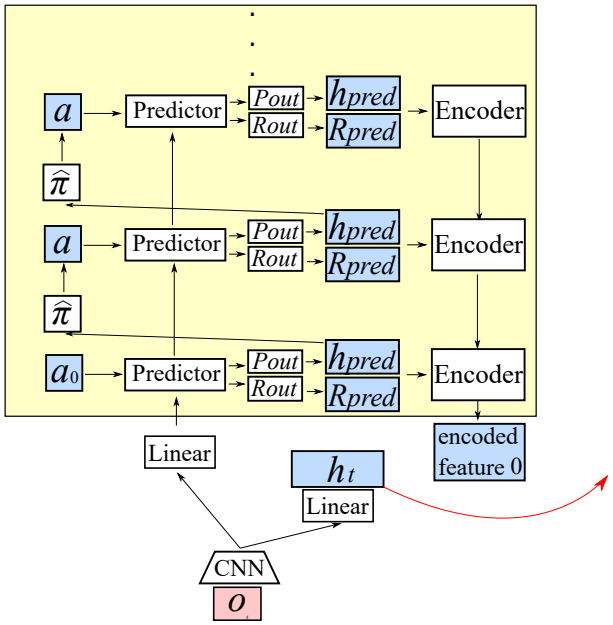

図 1.4: 報酬予測機構を追加した Predictor

な環境モデルの適切な取扱いと，軽量な計算コストを同時に実現するとともに，ゲームの状態を表すタスク達成に適した中間表現の獲得，中間表現をもとにした環境の状態遷移の学習，得られた環境モデルをもとにした方策の学習をすべて end-to-end で効率的に行うことができる．

アーキテクチャの概要は I2A [15] と類似しているが，低次元のベクトル表現を用いて環境モデルを学習するため，計算時間の観点で I2A よりも優れている．この点は次節の評価実験でも確認をおこなう．また，可能な全ての行動に対してロールアウトを行う点は RL-HLP [16] に倣っている．

モデルフリーなネットワークにより得られたベクトルを予測により得られた特徴量に結合する点については I2A の時点で行われていたが，I2A においてはモデルフリーな特徴量は方策決定を補佐する役割を持つのみであった．本提案手法ではこの特徴量を環境モデルを学習する際の中間表現として用いている．

SSM [18] や RL-HLP などの，抽象化された中間表現を用いる既存の手法と比較すると，提案手法は中間表現の事前学習が不要であるという点が異なっている．これにより，方策の学習が進み環境から得られる観測の分布が変化しても自動的に適切な中間表現が学習されることが期待できる．

CRAR [20] は，モデルフリーな強化学習により得られた特徴量の空間で環境モデルの学習を同時に行うという点で提案手法と類似している．しかし CRAR では強化学習の学習時においてはあくまでもモデルフリーかつ off-policy な DQN を用いており，環境モデルの学習は特徴表現獲得の補助と

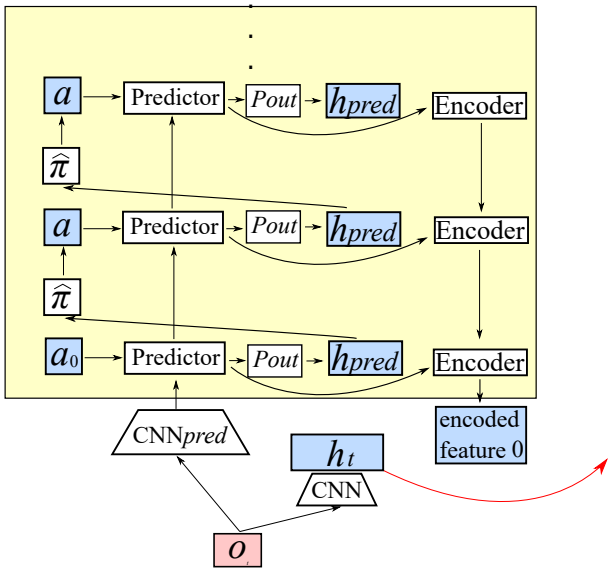

図 1.5: Predictor の隠れ層を Encoder に入力するモデル

表 1.1: 観測の抽象化を行うモデルベース強化学習の比較

	Task-Specific	Joint Learning	Abstracted Training Data
RL-HLP	○	×	○
World Model	×	×	△
SSM	△	×	×
CRAR	○	△	○
Proposed	○	○	○

して利用されている．本提案手法は環境モデルを用いたモデルベースな方策を同時に学習できるという点で CRAR と異なっている．

各手法の比較を表 1.1 に示した．Task-Specific はタスク達成に適した中間表現を使用しているか否かを表す．SSM は報酬を考慮しないが，状態遷移を考慮する点において World Models よりも優れているため，△とした．Joint Learning は方策と中間表現および状態遷移の学習を同時に行うことができるかどうかを示している．CRAR では強化学習と状態遷移モデルの学習を通して中間表現を得るが，強化学習がモデルフリーの DQN であり，状態遷移を考慮したものではないため△とした．Abstracted Training Data は，中間表現の学習や中間表現を用いた環境モデルの学習時に，誤差を計算する際のターゲットとして用いるデータがそのままの観測データに比べて低次元のものか否かを表す．RL-HLP では，中間表現は A3C の強化学習により獲得し，状態遷移も事前学習した中

間表現のみの遷移を扱うため，高速に計算可能である．World Model では，中間表現の獲得に VAE を用いるため，誤差関数として高次元の観測同士の距離を計算する必要がある．しかし状態遷移の学習時には中間表現を固定して，中間表現同士の誤差計算を行う．SSM は，中間表現の獲得と環境モデルの学習を同時に行うが，その際実際の観測をターゲットとして用いるため，計算量が大きい．CRAR および提案手法は強化学習による中間表現の獲得と中間表現同士の誤差計算による環境モデルの学習を行うため，高速に計算が可能である．

4.2　倉庫番ゲームにおける評価実験

提案手法の有用性を評価するため，倉庫番ゲームを用いた評価実験を行い，モデルフリーな手法および I2A と比較した．

4.2.1　倉庫番

倉庫番とは，グリッド上の環境内でキャラクターを移動させて箱を押すことで，全ての箱を目標位置に移動させることが目的のゲームである．キャラクターは箱を押すことはできても引くことはできないため，先を見据えた計画が必要となる．I2A の論文においても，本研究で行った倉庫番ゲームと同様のゲームで実験が行われていた．今回の実験では先行研究 [15] に倣い，図 2.1 に示したような画像をエージェントの入力とした．緑色のキャラクターを上下左右の四方向に移動させ，全ての茶色い箱を赤い目標位置の上に載せるとクリアとなる．今回の実験では，最大 8×8 のグリッド中に，4 つの箱を配置する問題を 10^7 問自動生成し，その中からランダムに出題することとした．また，120 ステップ経過してもクリアできなかった場合は，不正解として次の問題に進むようにした．ゲームの報酬設計も先行研究 [15] と同様に，毎ステップ -0.1，箱を目標位置においた際に 1，箱を目標位置から外した際に -1，ゲームクリア時に 10 の報酬が入るようにした．

各グリッドは 8×8 ピクセルの画像で構成されており，周囲に必ず壁となるグリッドが 1 マスずつ配置されるため，10 グリッド ×10 グリッド分，すなわち 80 ピクセル ×80 ピクセルの画像が観測として得られる．

4.2.2　モデルの概要

I2A を除く全ての手法において，畳み込みニューラルネットワークにより 512 次元の中間表現 h を得る．ここでの畳み込みニューラルネットワークは，まず 80×80 ピクセル，3 チャンネルの観測に

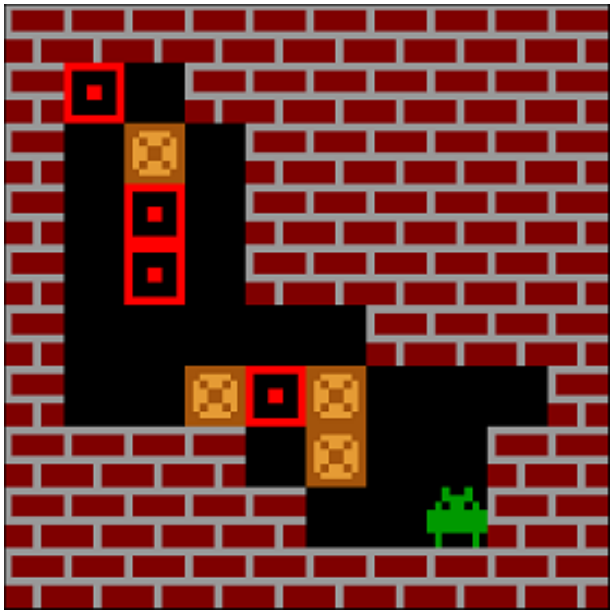

図 2.1: 倉庫番ゲームにおける観測の一例

対し，カーネルサイズ 8，ストライド 4，出力 32 チャンネルの畳み込み層を適用し，続けてカーネルサイズ 4，ストライド 2，出力 64 チャンネル，カーネルサイズ 3，ストライド 1，出力 64 チャンネルの畳み込み層を適用する．その後，出力サイズ 512 次元の全結合層を適用して h を得た．各層の出力には活性化関数として Leaky ReLU を適用した．モデルフリーの手法は h を直接全結合ネットワークに入力し，価値関数と方策関数の値を得る．提案手法では，行動を 512 次元のベクトル表現とし，Predictor 及び Encoder の入力サイズ，隠れ層サイズは全て 512 とし，状態の先読みステップ数は 5 とした．ただし，報酬の予測機構を追加したモデルについては，先行研究 [15] と同様に 4 通りの報酬を softmax で予測する機構を追加した．Encoder への入力に関してはその 4 次元の確率分布と 512 次元の h を結合するため，このモデルについてのみ 516 次元となる．

また，パラメタサイズに対する性能を評価するため，畳み込み層のチャンネル数と隠れ層 h の次元を全て二倍にしてパラメタサイズを提案手法に近づけた model-free エージェントについてもテストを行った．

学習には IMPALA [19] を用い，アドバンテージ計算のためのステップ数は 12 とした．Optimizer としては RMSprop [37] を使用した．

各モデルのパラメータ数を表 2.1 に示した．RL は強化学習による価値関数と方策関数の誤差で学習されるパラメータ数であり，prediction は予測に用いられるパラメータ数である．ロールアウト

表 2.1: 各モデルのパラメタ数

Model	RL	Prediction
Model-free	1.26×10^6	N/A
Proposed (Simple)	2.84×10^6	1.58×10^6
Proposed (Separate)	2.84×10^6	3.10×10^6
Proposed (Separate + Shared conv)	2.84×10^6	3.10×10^6
Proposed (Separate + Shared conv + Reward prediction)	3.10×10^6	3.36×10^6
Proposed (Separate + Shared conv + Use hidden of Predictor)	2.84×10^6	3.10×10^6
I2A	6.02×10^6	1.32×10^6
Model-free (Large)	5.02×10^6	N/A

ポリシーに関するパラメータは prediction に含むものとした．なお，畳み込み層を共有するモデル（Shared conv モデル）については，RL と Prediction で 7.6×10^4 個のパラメータを共有している．

4.2.3　結果

学習を行った結果，解くことができた問題の割合を図 2.2 に示す．一定ステップ学習する毎に 10^4 問の問題を与えて正答率を確かめたものである．なお，I2A でも実験を行ったが，表 2.2 に示すように計算が非常に遅く，結果を得ることができなかった．Proposed (Simple) を除く各モデルについて 2 回ずつ実験を行い，平均をとった．Proposed (Simple) はモデルフリーの結果を下回っている．これについては，中間表現 h のみから次状態の中間表現を予測することは難しく，予測により有用な情報を得ることができなかったと考えられる．これに対し Proposed (Separate) 及び Proposed (Separate + Shared conv) はモデルフリーエージェントの結果を大きく上回っている．また Proposed (Separate + Shared conv) の方がよりサンプル効率が良いという結果が得られた．なお，Proposed (Separate + Shared conv + Use hidden of Predictor) は Proposed (Separate + Shared conv) と比較して学習が進み始めるまでが遅く，また最終的なスコアにはほとんど差がなかった．これは学習初期は Predictor の隠れ層が大きく変化するため Encoder が特徴抽出を学習するのに時間がかかるなどの原因が考えられる．

4.2.3.1　学習時間の評価

各手法の学習速度を比較した結果を表 2.2 および図 2.3 に示した．Proposed (Simple) と I2A を除く各モデルについて 2 回ずつ実験を行い，平均をとった．なお，実験環境は 6 コア 12 スレッドの

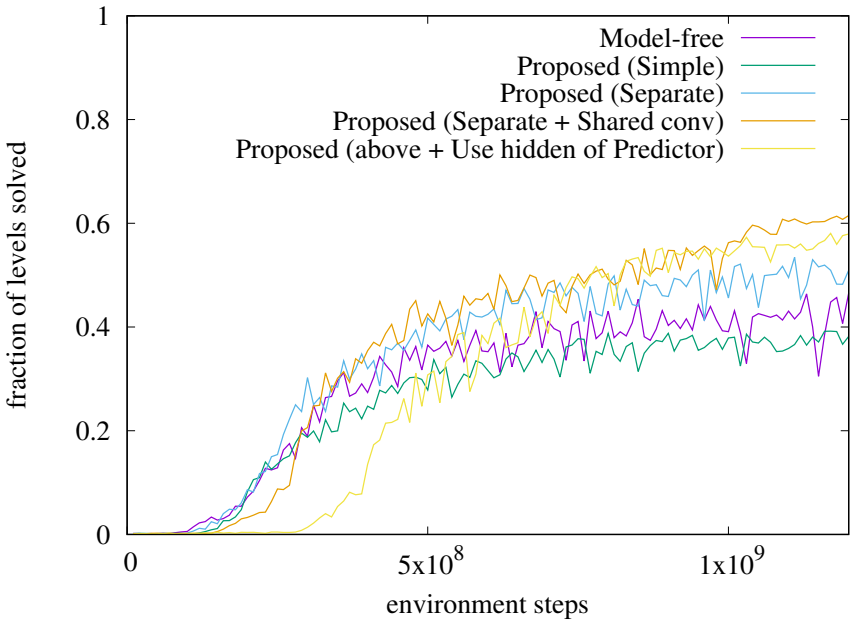

図 2.2: エージェントが経験した状態数に対する性能の評価

コア数を持つ CPU 及び単一の GTX 1080 Ti を備えている．提案手法はシンプルなモデルフリーな学習の 3 倍程度の計算コストで学習ができている事がわかる．今回の実験では環境が非常に低コストで計算可能であり，GPU 上でのニューラルネットワークの計算のみが計算時間に影響を与えているが，環境の状態遷移の計算コストがより大きな環境においては，提案手法とモデルフリー手法の計算時間の比率はより小さくなると予想される．I2A については，[15] で行われていたように，環境モデルを事前学習した場合の計算時間と，本稿の提案手法と同様に環境モデルを同時学習する場合の計算時間を調べた．提案手法は I2A の 1/10 程の時間で学習可能であることがわかる．なお，I2A は学習時に多くの GPU メモリを必要とするため，他の手法と同様のバッチサイズで学習することができなかった．比較のため，提案手法を I2A と同様のバッチサイズで学習した際の速度も Proposed (small batch size) として併記した．

正答率と学習時間の関係を図 2.4 に示した．モデルフリーな手法は高速に学習可能なため，初めは提案手法を上回っているが，20 時間ほど経過した後に提案手法の正答率が上回る様子が確認できる．

表 2.2: 各モデルの学習時間

Model	Training speed (Steps/s)
Model-free	2.68×10^4
Proposed (Simple)	9.92×10^3
Proposed (Separate)	8.72×10^3
Proposed (Separate + Shared conv)	1.000×10^4
Proposed (Separate + Shared conv + Reward prediction)	8.41×10^3
Proposed (Separate + Shared conv + Use hidden of Predictor)	1.000×10^4
I2A	8.07×10^2
I2A (with env model learning)	6.44×10^2
Proposed (Separate + Shared conv, small batch size)	6.54×10^3

4.2.3.2　報酬予測機構の評価

報酬予測機構による性能の変化を確認するために，他の手法と比較した結果を図 2.5 に示す．報酬の情報を追加したにもかかわらず，学習が遅くなり，最終的な結果も向上しなかった．これに関しては，報酬の予測により Predictor の学習がより難しくなったことが原因だと考えられる．報酬 $r(s_t, a_t, s_{t+1})$ は状態遷移の際に発生するものである．報酬を予測しない場合は GRU [36] の隠れ層は状態遷移後の状態 s_{t+1} に対応する中間表現を出力するのに必要な情報のみを保持すれば良いのに対して，報酬を予測する場合はどのように状態遷移したかの情報も表現しなければならない．これにより Predictor の学習が遅れ，学習の遅れにつながったのではないかと考えられる．

4.2.3.3　パラメタサイズに対する性能の評価

パラメタサイズを大きくしたモデルフリーなエージェントのスコアに対する比較を図 2.6 に示す．パラメタサイズの大きなモデルフリーエージェントは提案手法よりもやや学習が早く，最終的な結果はほとんど変わらないことが確認できる．提案手法は環境モデルを用いた探索による情報を用いているが，表 2.1 に示すように，Model-free (Large) は強化学習の誤差により学習可能なパラメタが多いため，学習初期から比較的高いスコアを出すことができたのだと考えられる．

4.2.3.4　予測に用いる行動の選択方式に対する評価

本提案手法では，RL-HLP [16] と同様に，現在の観測から予測を行う際は可能な全ての行動に対して次状態を計算し，その後ロールアウトポリシーに従ったロールアウトを行う．これに対し I2A

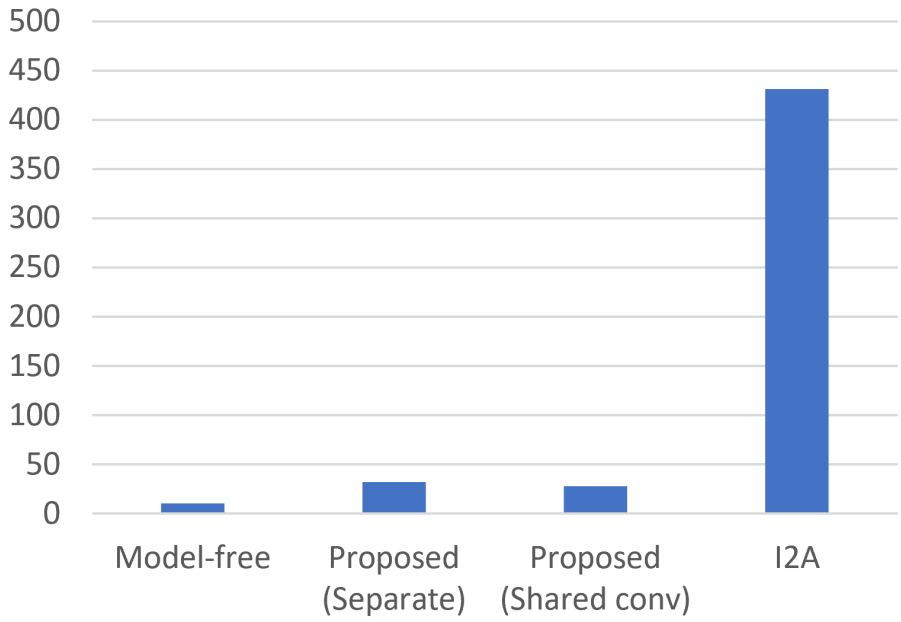

図 2.3: 10^9 ステップの学習に要する時間（Hours）

では現在の観測をロールアウトポリシーに入力し，方策からのサンプルにより複数回のロールアウトを行う．これらの手法を比較するため，提案手法において I2A と同様に最初の行動をロールアウトポリシーからサンプルした場合の性能評価を行い，結果を図 2.7 に示した．なお，今回の実験において可能なアクションは 4 通りであり，通常の手法では 4 通りのロールアウトを行っていたため，本実験ではロールアウトポリシーから 4 つの行動をサンプルし，4 回のロールアウトを行った．図 2.7 の Proposed (Reward prediction) 及び Proposed (Sample first actions) はいずれも畳み込み層の共有と報酬予測機構の追加を行っている．I2A と同様のサンプルを行う方式は RL-HLP と同様に全ての行動に対してロールアウトを行う方式よりも低いスコアとなった．しかし行動空間が連続であったり，非常に多くの可能な行動が存在する場合は全ての行動に対してロールアウトを行うことは不可能であるため，サンプルする方式を用いることも考えられる．

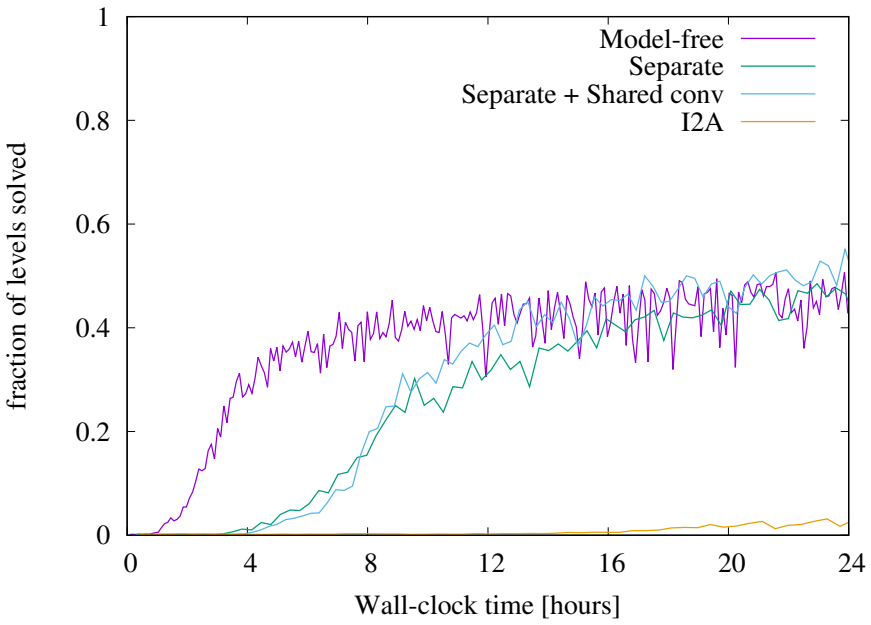

図 2.4: 学習時間に対する性能の評価

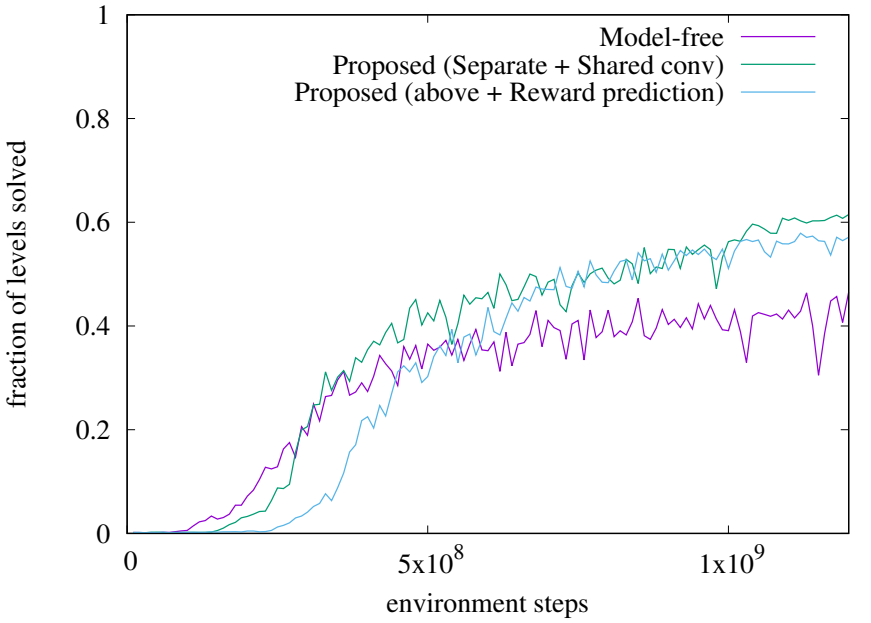

図 2.5: 報酬予測機構の評価

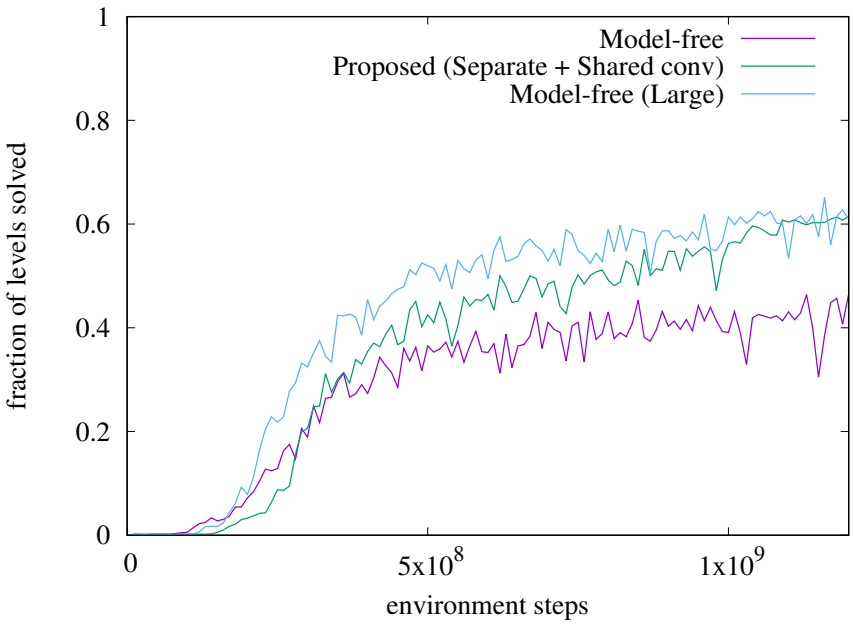

図 2.6: パラメタサイズに対する性能の評価

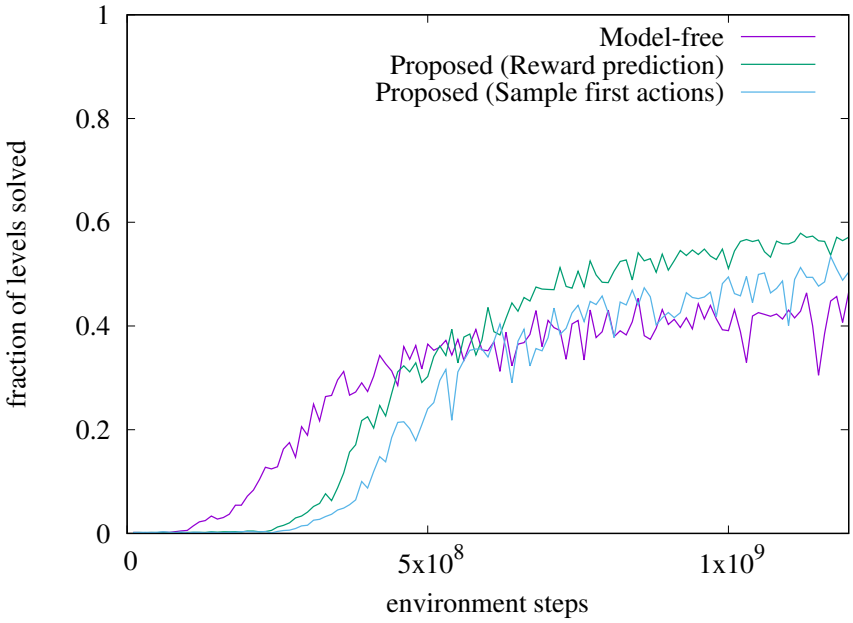

図 2.7: 予測に用いる行動の選択方式に対する評価

第5章　結論

5.1　まとめ

本稿では，タスク達成に適した低次元のベクトル表現を用いて次状態の予測を行う新しいモデルベース強化学習のアーキテクチャを提案した．モデルベース強化学習において，観測が高次元であるような環境モデルの学習の計算コストを削減するために，抽象化した中間表現を用いることが有用である．しかし強化学習においては，方策の学習が進むにつれて観測の分布が変化することがあり，中間表現の事前学習を行うと性能が劣化する恐れがある．そのため提案手法では，環境モデルを用いた予測により得られる特徴量にモデルフリーな特徴量を結合して中間表現として用いることにより，ゲームの状態を表すタスク達成に適した中間表現の獲得，中間表現をもとにした環境の状態遷移の学習，得られた環境モデルをもとにした方策の学習を同時に行うことを実現した．

倉庫番ゲームにおける評価実験にて，モデルフリーな手法以上のスコアを達成し，また既存のモデルベース強化学習のアーキテクチャである I2A と比べて極めて小さな計算コストで学習が可能であることを確認した．また，様々な提案手法を比較した結果，次状態予測を行う GRU の隠れ層を得るための CNN と中間表現 h を得るための CNN において畳み込み層を共有するアーキテクチャが，高速に計算可能かつサンプル効率がよいことが確認された．

5.2　今後の課題

今後の課題は，提案したアーキテクチャの性能向上が挙げられる．本稿の序論では性能という言葉について，学習が収束した際に得られる方策の精度，学習のサンプル効率および学習時やテスト時の計算時間の削減といった要素を挙げたが，例えば学習が収束した際に得られる方策の精度やサンプル効率の改善のためにまず考えられるのはネットワーク構造の改良である．今回の実験においてはアーキテクチャの各要素について性能に対する寄与度を細かく調査しなかったが，それらを調査することで，重要なモジュールを割り出し，改良することで性能向上の見込みがある．また，逆に寄与度の小さいモジュールの省略，簡易化によって計算時間をさらに削減することも可能となると思われる．

また，既存手法との比較実験も重要であると考えられる．上で述べたこととも関連するが，各要素の性能に対する寄与度が不明なため，どのような要素により既存手法との差異が出ているかが依然として判明していない．構成要素を整理し，既存手法と要素の組み替えおよび比較を行うことで，重要な点を割り出すとともに，より性能を高めることができると期待される．

本稿の実験においては，報酬を予測するネットワークは学習を遅くする結果を招いてしまった．これは実験の項でも述べたように，報酬は状態遷移の際に発生するものであり，報酬を予測しない場合は GRU [36] の隠れ層は状態遷移後の状態に必要な情報のみを保持すれば良いのに対して，報酬を予測する場合はどのように状態遷移したかの情報も表現しなければならないことに起因すると考えられる．これを解決するために，GRU の構造をそのまま用いるのではなく，GRU における次状態の計算時に報酬に相当する値を予測する機構を追加するなどの変更が考えられる．

学習して得られた環境モデルを活用することで性能の向上を図ることも可能である．モデルベース強化学習においては，学習した環境モデルを用いて仮想的にゲームをプレイし，得られた経験から強化学習を行うことでサンプル効率をより改善する手法が提案されている [17]．また，好奇心ベースの手法の一部 [25] は，現在の状態と行動の組から次状態が予測できなかった場合に，エージェントにとって未知の状態であるとしてエージェントに報酬を与える．これを本提案手法の環境モデルの学習と組み合わせることで，比較的少ないオーバーヘッドで効率の良い探索を行うことが可能なアーキテクチャを実現することが期待できる．

今回は先行研究 [33] と同様に，次状態の中間表現を予測するにあたって，学習時の誤差関数は Huber Loss を用いたが，今回用いた中間表現は非線形関数であるニューラルネットワークの中間層であり，単純な二乗誤差が小さいことは必ずしも状態が類似していることを示さない．そのような不正確性も含めた上で Encoder が特徴抽出法を学ぶことを期待しているのは確かだが，より状態を正しく表す予測が可能であることが望ましい．近年では，転移学習の文脈において隠れ層のユニットが活性化する境界を用いて隠れ層同士の比較および学習を行う手法 [38] なども提案されており，中間表現同士の比較法は十分検討の余地があると考えられる．

アーキテクチャの適用範囲を広げるという点での改善も考えられる．本稿では状態を完全に観測可能であり，状態遷移が確定的であるようなマルコフ決定過程のみを対象とした．しかし毎周期状態の完全な観測が得られない部分観測マルコフ決定過程の問題も多く存在する．部分観測マルコフ決定過程における最適方策を得るためには現時点での観測だけではなく，過去の観測を記憶しておく必要がある．これに対応するために，リカレントニューラルネットワークを用いて過去の観測を含めて中間表現にエンコードすることが考えられる．また，環境によっては状態遷移が確率的である

ことも考えられる．確率的な状態遷移を学習するために，MDN-RNN [31, 17] などの手法と組み合わせることが考えられる．また，本稿では行動が離散かつ比較的少ない種類のみが存在する場合を扱ったが，連続な行動空間や多くの選択肢をもつ行動空間を扱う場合は次状態予測の際に全ての可能な行動について予測を行うことが不可能である．そのため実験の項でも述べたように I2A と同様の，最初の行動を選ぶ際にもロールアウトポリシーを用いてサンプルするアーキテクチャを用いることが考えられる．しかし方策を決定するという点においては様々な行動について試行することは有用であると考えられるため，巨大な行動空間においても効率よく様々な行動に対するロールアウトを行うアーキテクチャの考案が望まれる．

参考文献

[1] Michael Buro. Experiments with multi-probcut and a new high-quality evaluation function for othello. *Games in AI Research*, pp. 77–96, 1997.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *NIPS Deep Learning Workshop*, 2013.

[3] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pp. 2094–2100, 2016.

[4] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *ICML Deep Learning Workshop*, 2015.

[5] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, Vol. 1. MIT press Cambridge, 1998.

[6] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in neural information processing systems*, pp. 1–8, 2007.

[7] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, Vol. 4, No. 26, p. eaau5872, 2019.

[8] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforce-

ment learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1928–1937, 2016.

[9] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, Vol. 47, pp. 253–279, 2013.

[10] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, Vol. 550, No. 7676, pp. 354–359, 2017.

[11] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings 1990*, pp. 216–224. Elsevier, 1990.

[12] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv:1611.05763*, 2016.

[13] David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 3191–3199, 2017.

[14] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *Advances in Neural Information Processing Systems*, pp. 6114–6124, 2017.

[15] Sébastien Racanière, David Reichert, Theophane Weber, Oriol Vinyals, Daan Wierstra, Lars Buesing, Peter Battaglia, Razvan Pascanu, Yujia Li, Nicolas Heess, et al. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 5692–5699, 2017.

[16] Hirotaka Kameko, Jun Suzuki, Naoki Mizukami, and Yoshimasa Tsuruoka. Deep reinforcement learning with hidden layers on future states. *Computer Games Workshop at IJCAI*, 2017.

[17] David Ha and Jürgen Schmidhuber. World models. *arXiv:1803.10122*, 2018.

[18] Lars Buesing, Theophane Weber, Sebastien Racaniere, SM Eslami, Danilo Rezende, David P Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv:1802.03006*, 2018.

[19] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[20] Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. *arXiv:1809.04506*, 2018.

[21] Hamid R Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 719–726, 2010.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, Vol. 518, No. 7540, pp. 529–533, 2015.

[23] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[24] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

[25] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[26] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv:1810.12894*, 2018.

[27] Alfredo V Clemente, Humberto N Castejón, and Arjun Chandra. Efficient parallel methods for deep reinforcement learning. *arXiv:1705.04862*, 2017.

[28] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Reinforcement learning thorugh asynchronous advantage actor-critic on a gpu. In *ICLR*, 2017.

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

[30] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[31] David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv:1704.03477*, 2017.

[32] Razvan Pascanu, Yujia Li, Oriol Vinyals, Nicolas Heess, Lars Buesing, Sebastien Racanière, David Reichert, Théophane Weber, Daan Wierstra, and Peter Battaglia. Learning model-based planning from scratch. *arXiv:1707.06170*, 2017.

[33] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 4739–4748, 2018.

[34] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

[35] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv:1808.04355*, 2018.

[36] Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[37] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, Vol. 4, No. 2, pp. 26–31, 2012.

[38] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. *arXiv preprint arXiv:1811.03233*, 2018.

発表文献

本研究に関する発表文献は以下の通りである．

- 水谷陽太, 鶴岡慶雅. モデルベース深層強化学習における隠れ層を用いた環境遷移モデルの提案. ゲームプログラミングワークショップ 2018 論文集, pp. 72-79, 2018.
- Yota Mizutani and Yoshimasa Tsuruoka. Learning Task-Specific Representations of Environment Models in Deep Reinforcement Learning. AAAI-19 Workshop on Reinforcement Learning in Games, 2019.

また，本研究とは直接の関係はないが，修士課程中に発表した文献は以下の通りである．

- 水谷陽太, 鶴岡慶雅. 隠れマルコフモデルによる歴史テキストの人物移動のモデル化. 人工知能学会全国大会論文集, Vol. JSAI2018, pp. 4Pin1-17, 2018.

謝辞

本研究を進めるにあたり，非常に多くの方にお世話になりました．

指導教員である鶴岡慶雅教授には研究に関する様々な面でお世話になりました．私が研究室に配属された際の指導に始まり，研究テーマの決定，研究を進める上でのアドバイス，関連する論文の紹介，論文執筆時の構成，英語の書き方など，とても挙げきれないほどのご指導をいただきました．感謝してもしきれません．この研究を始めて，ここまで続けることができたのは間違いなく先生のおかげです．また研究のみならず，先生の人柄のおかげで修士生活を非常に楽しく過ごすことができました．本当にありがとうございました．

私を今まで支えてくださった家族には心の底から感謝しております．私の主体性を重視していただき，適切な距離感で私を支えてくれて，やりたいことをやれる環境を用意してくれたことで，大変充実した生活を送ることができました．このような家族を持つことができたことを誇りに思います．

研究室の同期である河村圭悟君とは，研究についての話だけでなく，技術に関する話題，趣味，数学の問題などについて大変楽しい議論を交わすことができ，本当に感謝しております．この研究分野について理解を深めることができたのは河村君の影響が非常に大きいと感じています．河村君の話はとても面白く，研究に関することもそうでないことも，確実に今の私を形成する要素の一部となっていることを感じます．また，私の話に対しても興味を持って聞いてもらえて非常にうれしかったです．私の研究生活における MVP は河村君です．

私が修士一年のときには，当時博士過程であった亀甲博貴さんに様々な面でお世話になりました．研究分野に関する知識から，研究テーマに関する相談，研究室生活についてなど，有意義なアドバイスを多数いただき，大変感謝しております．また，そのほかの先輩方もたくさん話をさせていただき，研究についてや人生の送り方など，多くの面で参考にさせていただきました．本当にありがとうございます．

後輩の皆さんも，研究や趣味に関する雑談に付き合っていただき，研究分野に関する理解を深めることができました．気軽に会話をしてくれた研究室の皆さんのおかげで，本当に楽しい研究室生活を送ることができました．ありがとうございました．