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

水谷陽太[1,a)]　鶴岡慶雅[2]

概要：近年，深層強化学習の分野において，モデルベース強化学習に関する研究が注目を集めている．環境モデルを学習することで先読みを行うことが可能となり，長期的な計画に基づく方策決定が可能となる．ビデオゲームなどの複雑な環境において，画像を観測として用いる場合，観測を直接予測するような環境モデルを学習するのには莫大な計算コストがかかる．そのため，画像を低次元のベクトル表現に変換し，その表現における次状態の予測をする学習を行う手法がいくつか提案されている．しかし既存の手法における中間表現は，強化学習の目的においては不要なデータを含んでいたり，事前学習を必要とするなど，一長一短であった．本論文では，タスク達成に有用な中間表現を学習すると同時に，その中間表現を用いて次状態予測の学習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する．提案するアーキテクチャは，表現の獲得と次状態予測，方策決定の全てを end-to-end で効率的に学習ができ，比較的小さな計算コストで長期的な計画に基づく方策決定を行うことが可能である．実験により，ビデオゲームの深層強化学習において提案手法を用いることで効果的な学習を行えることを確認した．

Introducing an Environment Model based on Hidden States for Model-Based Deep Reinforcement Learning

Yota Mizutani[1,a)]　Yoshimasa Tsuruoka[2]

Abstract: Today, research about model-based deep reinforcement learning is attracting a lot of attention. These algorithms allow agents to predict future states and plan for the future. With complex environments such as video games, if raw images are used as observations, calculation cost for predicting the next observation becomes too high. To deal with this problem, there are some methods to convert images into a low dimensional vector, and predict the vector corresponding to the next state. However, these approaches have some disadvantages. For example, some methods need pre-training of representation, or representation of some methods are not optimized for solving the task. In this paper, we introduce a new model-based reinforcement learning architecture using hidden states which are useful for the task. The architecture learns suitable representation, how to predict the next state, and how to decide actions simultaneously and efficiently. The architecture plans for the future with low calculation cost, and achieves a higher score than an existing model free architecture in a video game experiment.

1. はじめに

近年，Deep Q-Network (DQN) [1] をはじめとして，深層学習と強化学習を組み合わせた深層強化学習により，ビデオゲームの画面情報を直接観測として用いるゲーム AI の研究が盛んに行われている．深層強化学習によるビデオゲーム AI の研究においては，DQN や Asynchronos Advantage Actor-Critic (A3C) [2] を元にした手法が用いられる場合が多いが，その殆どにおいてモデルフリーとよばれる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（ゲーム）のモデルを意味する．すなわち，

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

環境に対して行動を行った際に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．例えばDQNでは，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値を経験から学習する．この値をQ値と呼ぶ．ゲーム中のあらゆる状態において正確なQ値を学習すると仮定すると，モデルフリー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環境において，全ての状態におけるQ値を完全に学習することは不可能に近い．これは主に環境の状態数が非常に大きく，ニューラルネットワークの表現力が不足していることや，表現力を増すと現実的な時間で学習が収束しないことなどによる．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は将来の状態を見据えて長期的な計画に基づいた行動をとることを苦手とする傾向にある．

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な状態遷移を行いより探索をすることで，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．ビデオゲームや現実世界におけるタスク等の複雑な環境においては，環境のモデルが未知であることが多いため，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 [3]．しかし，モデルベース深層強化学習にはいくつかの課題が存在する．一つ目は，ゲーム画面などの高次元のデータを環境からの観測として用いているため，それをそのまま環境のモデルとして定義してしまうと，環境のモデルを学習するニューラルネットワークの入出力が高次元となってしまう点である．これにより学習時やテスト時における計算コストが大きくなり，適用範囲が限られてしまう．二つ目の問題点は，複雑な環境モデルを完全に学習することは困難なため，環境のモデルが不完全になってしまう点である．不完全に学習された環境で繰り返し次状態の予測を行うと，誤差が蓄積し逆に性能を悪化させてしまう．これらの課題を解決するための手法がいくつか提案されているが [4][5][6][7]，どれも一長一短であり，上記の課題を同時に解決するには至っていない．

本論文では，モデルベース強化学習の新しいアーキテクチャを提案し，実験により提案手法の有用性を確認した．提案するアーキテクチャでは，畳み込みニューラルネットワークを用いて観測を低次元のベクトル空間へと変換したものを中間表現とし，中間表現の空間で予測した次状態をLong Short-Term Memory (LSTM) [8] を用いてエンコードして最終的な方策の決定に用いる．本手法により，タスク達成のために有用な隠れ層の表現を獲得すると同時に，隠れ層空間で次状態を予測し，さらに予測された次状態を適切に用いて方策の決定を行う方法を学習することができる．これにより，高次元の入力を持つ環境においても比較的小さな計算コストで長期的な戦略を立てることが可能となった．

2. 関連研究

2.1 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 [6]．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．

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

これ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．従来のモデルベースの手法とは異なり，予測された状態を用いて木探索などを行わずに，途中の状態を全て結合し，ニューラルネットワークによって方策決定を行っている点が特徴である．探索を行う場合は，より先のステップの値を用いることで先を見据えた長期的な方策決定ができる反面，不完全な環境モデルにおいて多くのステップで探索を行うと予測誤差が積算し，性能の劣化を引き起こすという問題がある．そこで，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，予測された各状態の信頼度を含めて総合的に判断して方策決定を行うようにする学習が可能となる．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．動的に先読みの方法を変更する研究は既に [9] などで行われている．

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．

2.2 Imagination-Augmented Agents (I2A)

Weber らは 2017 年，モデルベース強化学習の新たなアーキテクチャとして Imagination-Augmented Agents (I2A) を提案した [5]．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，LSTM [8] 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，環境モデルによる予測を n ステップ先まで繰り返し，予測された観測及び報酬を，時系列の逆順に LSTM を用いたエンコーダに入力していく．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エージェント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．RL-HLP と同様にニューラルネットワークによってモデルの正確性も含めて最適な方策の学習を行うが，RL-HLP とは異なり，途中の状態を結合せずに順に LSTM に入力することで，予測ステップ数の変化に柔軟に対応することが可能である．また，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，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．画像の情報量が増えるに従って計算コストも上がるため，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．

2.3 World Model

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 [7]．RL-HLP と同様に，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ベクトル表現を用いる．RL-HLP は事前学習した A3C のエージェントの畳み込み層を利用したのに対し，World Model は環境から観測として得られた画像を Variational Autoencoder [10] でエンコーディングする．すなわち，RL-HLP ではタスクを達成する上で必要な情報を得るための畳み込みニューラルネットワークでベクトル表現を得ていたが，World Model では画像の特徴量を表すベクトル表現を用いる．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という欠点がある．

World Model では状態遷移モデルに MDN-RNN [11] 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．

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．

2.4 中間表現の獲得

次状態を予測するために，高次元の画像情報を低次元の中間表現に落とし込む研究は複数存在する．

RL-HLP では，事前学習した A3C のエージェントの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) [12] 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 [13] 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 [14]．

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

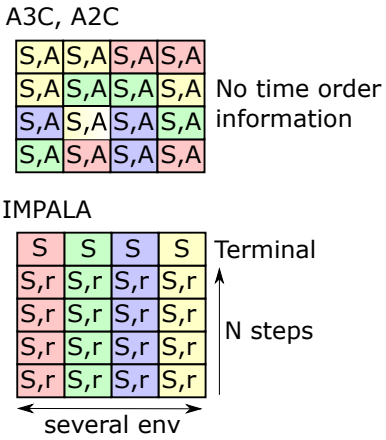

図 1 Training batch of IMPALA. S means an observation, A means pre-calculated advantage, and r means rewards.

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．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．

2.5 IMPALA

Espeholt らは 2018 年，分散型強化学習のアーキテクチャとして Importance Weighted Actor-Learner Architecture (IMPALA) を提案した [15]．IMPALA では，多数の Actor を並列で実行し，行動を決定するために必要な観測データ及び生成された学習データをそれぞれキューに積み，キューからバッチを作成して順次 Learner 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 [16] 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 Actor によって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．この方策のずれに対応するため，IMPALA では importance sampling を行うとともに，advantage の計算を学習時まで遅延する手法をとっている．通常 A3C においては，行動時に次状態の価値関数の値を取得し，それを用いて advantage を計算する．IMPALA においては，学習データの生成時に次状態の価値関数を計算してしまうと，学習時までに価値関数の値が変化してしまうため，図 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 advantage の計算を行っている．Advantage 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．

アルゴリズム 1 policy and value function

s 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
$f \leftarrow \mathrm{concat}(h^0, encoded_0, encoded_1, \cdots, encoded_n)$
$\pi \leftarrow \mathrm{softmax}(W_\pi f + b_\pi)$
$V \leftarrow W_V f + b_V$

アルゴリズム 2 policy and value function (full model)

s 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
$f \leftarrow \mathrm{concat}(h, encoded_0, encoded_1, \cdots, encoded_n)$
$\pi \leftarrow \mathrm{softmax}(W_\pi f + b_\pi)$
$V \leftarrow W_V f + b_V$

3. 提案モデル

本論文で提案するアーキテクチャでは，画像をタスク達成に適した中間表現に変換し，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 LSTM [8] を用いてエンコードして最終的な方策を決定する．提案手法により，これらの学習を同時に効率よく行うことができる．

アーキテクチャの概要を図 2 及びアルゴリズム 1 に示す．まず，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 s を特徴ベクトル 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 LSTM の入力とし，LSTM 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 LSTM 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 LSTM に入力し，エンコー

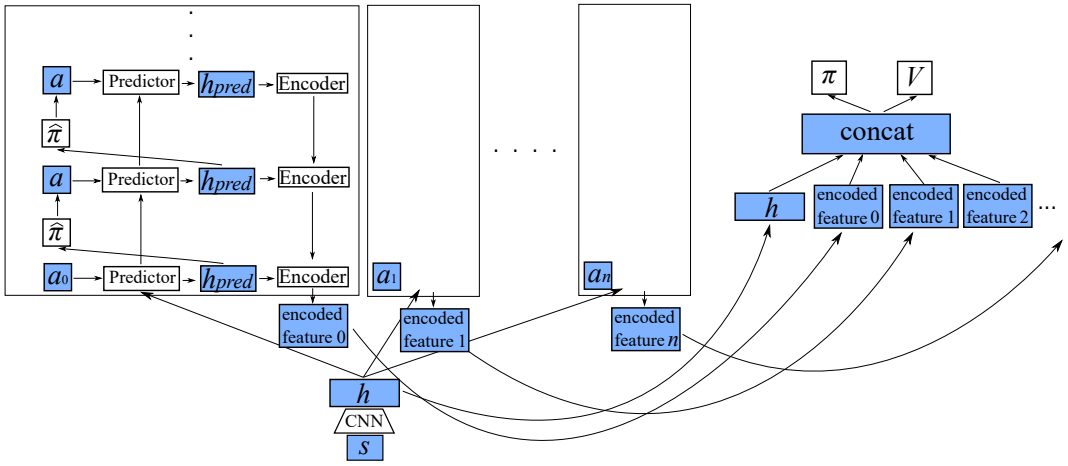

図 2 Overview of proposed architecture

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 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 が返す次状態は不正確なものであるが，方策決定においてはもとの隠れ状態 h を結合して用いているため，h をもとに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 が良い中間表現になるとともに，Predictor の精度も向上していくため，Predictor の出力結果をもとに 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できる．

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 [12] 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 [15] 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．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．

ロールアウトポリシー $\hat{\pi}$ に関しては，I2A [5] と同様に，

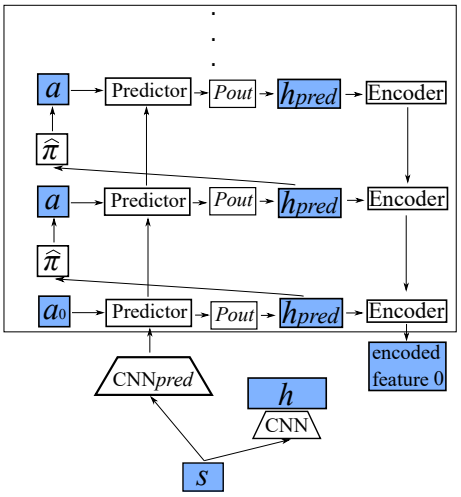

図 3 Predictor of full model

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．

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

また，h はあくまで価値関数及び方策関数を得るための表現であるので，環境によっては次状態の h を計算するために h の情報のみでは不十分になることが考えられる．そのため，より正確な次状態予測を可能とするために，図 3 に示すような予測器を用いることもできる．本稿では以後この予測器を用いるモデルを Full model と呼ぶ．Full 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 s 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．Full model のアルゴリズムをアルゴリズム 2 に示した．

提案手法は，I2A と同様に，予測された状態を LSTM 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

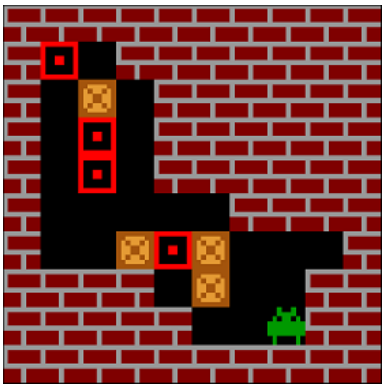

図 4　an example of the Sokoban game

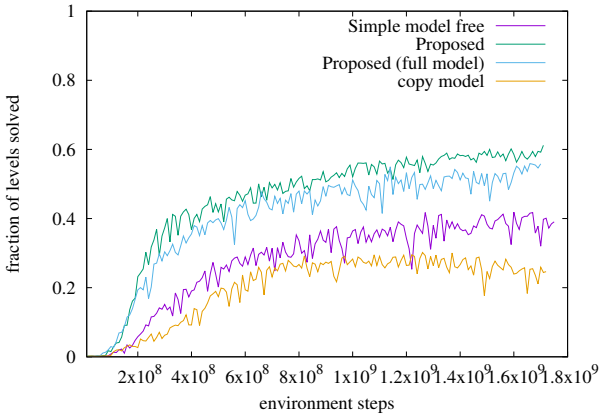

図 5　result of the experiment

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という点で，他の手法と異なっている．

4. 実験

提案手法の有用性を確かめるため，倉庫番ゲームにおいて，モデルフリーの手法及び提案手法にて学習を行い，結果を比較した．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 [5] 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 [5] に倣い，図 4 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 10^6 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 [5] 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．

表 1　Parameter size

model	RL	prediction	all
Simple model free	1.26×10^6	N/A	1.26×10^6
Proposed	3.37×10^6	2.10×10^6	5.47×10^6
Proposed (full model)	3.37×10^6	3.36×10^6	6.73×10^6
Copy model	3.37×10^6	N/A	3.37×10^6
I2A	4.63×10^6	1.30×10^6	5.92×10^6

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．

全ての手法において，先行研究 [5] と同様の畳み込みニューラルネットワークにより，512 次元の h を得る．ただし，活性化関数として Leaky ReLU を用いた．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．Proposed 及び Proposed (full model) 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．

Copy model は，Predictor が行動によらず入力された直前の h をそのまま返す，すなわち $h_{pred} = h$ としたモデルである．それ以外の Encoder などの学習法は提案手法と同一である．

各モデルのパラメータ数を表 1 に示した．なお，I2A は [5] における no reward I2A を基準としている．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．ロールアウトポリシーに関するパラメータは prediction に含むものとした．

学習には IMPALA [15] 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 [17] を使用した．

学習を行った結果，解くことができた問題の割合を図 5 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 に示すように学習が非常に遅く，結果を得ることができなかった．Copy model 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．モデルフリーの既存

表 2 Training speed

model	training speed (Steps/s)
Simple model free	2.6×10^4
Proposed	8.8×10^3
Proposed (full model)	7.3×10^3
I2A	8.9×10^2
I2A (with env model learning)	8.8×10^2

手法に関しては，パラメータ数の違いからか，学習が不安定であり，4 回の学習中 2 回において価値関数の値が発散してしまったため，正常に学習が完了した 2 回の結果を用いた．その他の手法に関しては学習中に値が発散することはなかった．結果を見ると，提案手法がモデルフリーの既存手法よりも高いスコアを記録している．また，copy model のスコアはシンプルなモデルフリーのモデルを下回っている．原因として，学習パラメータ数の増加により学習が遅くなってしまったことなどが考えられ，それに対して提案手法のスコアがモデルフリーな手法を上回っているのは単に学習パラメータ数が増加によるものではなく，次状態の予測によるものだと考えられる．また，今回の実験においては提案手法において full model と簡易 model の差異が小さかった．次状態予測に関する誤差は h の表現法に影響を与えないため，簡易モデルでは十分な予測精度が出ない可能性が懸念されたが，今回の実験ではそのようなことはなく，むしろ full model よりも学習が早いという結果になった．理由としては，倉庫番においては価値関数と方策関数を得るために，h は次状態の予測に十分な情報を含むような学習がなされるということが考えられる．また，表 1 に示すように，パラメータ数が小さいため，学習が早まったものと考えられる．

各手法の学習速度を比較した結果を表 2 に示した．なお，実験環境は 12 コア 24 スレッドの 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，[5] 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．

5. おわりに

本論文では，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．提案手法は，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効率よく行うことができる．実際に，倉庫番ゲームにおいてモデルフリーな手法よりも高いスコアを出すことに成功し，計算コストも比較的小さく抑えることができることを確認した．

本提案手法は，予測部分の計算量を抑えたことにより，複雑な手法と組み合わせても現実的な計算時間で学習を行える可能性がある．例えば，MDN-RNN [11] と組み合わせることで確率的な状態遷移を扱うことが考えられる．また，今回の状態遷移モデルでは扱わなかった報酬の予測を行うネットワークとの組み合わせも性能の向上に役立つ可能性がある．

参考文献

[1] 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.

[2] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1928–1937, 2016.

[3] 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.

[4] 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.

[5] 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.

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

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

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

[9] 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 preprint arXiv:1707.06170*, 2017.

[10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

2017.

[12] 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.

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

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

[15] 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.

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

[17] 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.

