深層強化学習による NetHack 攻略に関する研究

大貫 泰弘† 清 雄一† 田原 康之† 大須賀 昭彦†

†電気通信大学情報理工学域 〒182-8585 東京都調布市調布ヶ丘 1-5-1

E-mail: †ohnuki.yasuhiro@ohsuga.lab.uec.ac.jp, {seiuny, tahara, ohsuga}@uec.ac.jp

あらまし 深層強化学習の研究は，二人情報完全ゲームだけでなく，ビデオゲーム等の不完全情報ゲームにおいても行われている．近年，複雑で確率的なロールプレイングゲームであるローグライクゲームにおいても，深層強化学習の研究が注目されている．しかし，ローグライクゲームや，ゲームの複雑性を保持した環境では，ゲームを攻略したエージェントはない．本研究では，ローグライクゲームの一つである NetHack を基にした The NetHack Learning Environment の Score タスクを利用し，MuZero を基にしたエージェントと，独自の報酬を追加した Stochastic MuZero を基にしたエージェントを比較した．各エージェントを 4000 エピソード学習させ，100 エピソードの評価を行った．その結果，Stochastic MuZero を基にしたエージェントの方が優れていると示された．

キーワード 深層強化学習，ローグライクゲーム，MuZero，NetHack，NLE

1. はじめに

近年，深層強化学習 AI に関する研究は，囲碁や将棋などの二人完全情報ゲームだけでなく，Atari 2600 などのビデオゲームにおいても研究が行われている．この中で，深層強化学習 AI 研究の発展により，チェス,囲碁,将棋などの二人完全情報ゲームだけでなく，不完全情報ゲームである Atari 2600 においても一切の事前知識なしに人間や他のエージェントを凌駕する AI が生み出されている[1]. さらに近年, StarCraft II [2] や NetHack など，様々なビデオゲームにおいて深層強化学習 AI の研究が行われている．

その中で，複雑で確率的な環境を持つロールプレイングゲームであるローグライクゲームにおいても，深層強化学習 AI の研究が注目されている．ローグライクゲームを単純化した環境の場合，成功した深層強化学習 AI は存在する[3][4]．一方で，NetHack [5]などの既存のローグライクゲームを環境として用いた場合や，既存ゲームの複雑性を維持したまま変更したものを環境として用いた場合，ゲームを成功（攻略）した深層強化学習 AI はまだない[4][6][7]．そこで，ローグライクゲームを攻略する強化学習 AI を作成することが課題の一つである．

ローグライクゲームとは，1980 年に Michael Toy, Glenn Wichman により開発された，ダンジョン探索型のコンピュータロールプレイングゲームである Rogue というゲームに類似した性質を持つゲームの総称である．ローグライクゲームの特徴は次のとおりである．

プレイするたびにランダムに生成されるマップダンジョンを探索し,一般にターン制のシステムであり，プレイヤーが行動しない限りゲーム内時間は経過しない．また，プレイヤーは訪れたことのない場所のマップを見ることができないという部分観測性がある．さらに，一度エージェントが死ぬと最初からゲームをやり直す必要があり，操作キャラクタは定期的に食事を行わなければ餓死をし，ゲームが終了する．

NetHack[5][9]とは，1987 年に発表されたローグライクゲームの一つである．

NetHack の目的は，ダンジョンを下り，アミュレットを回収し，半神に昇格することである．そのため，プレイヤーは手続き的に生成された 50 以上の階層を下り，アミュレットを回収し，最終階層を突破しなければならない．NetHack の特徴は，次の通りである．

食料を食べすぎた場合，窒息死し，即死攻撃を受けることや致命的な状態（石化，首を絞められる）になることがある．また，職業（モンク，騎士など）や種族（人間，エルフなど），性別，属性（中立，善性，混沌）をゲーム開始時に選択可能である．さらに，マップ情報が保存されるため，一度訪れた層をもう一度訪れる際，マップは立ち去ったときの状態である．

NetHack では，各階層は手続き的に生成され，プレイヤーは複数回同じ状況に遭遇する可能性はめったにないため，エージェントに複雑性を提供する．図 1 は NetHack のプレイ画面の一例である．

図 1 NetHack のプレイ画面の例

本研究の目的は，複雑で確率的な環境において，強化学習 AI を学習させ，攻略を進めるエージェントを作成することである．今回，複雑で確率的な環境として，NetHack と呼ばれるゲームに基づいた NetHack Learning Environment (NLE) [6]のタスクの 1 つである score タスクを用いる．

本研究では，確率的な環境に対応していないエージェントと対応しているエージェントを比較し，NLE においてどちらの手法が良いか比較する．そこで，確率的な環境に対応していない MuZero [1]を基にしたエージェントと，NLE の score タスクに独自の報酬を加えた確率的な環境に対応している Stochastic MuZero [8]を基にしたエージェントについて，NLE の score タスクにおける報酬値やどの程度ダンジョンを降りることができたかを示すダンジョンレベルなどを比較する．

その結果，NLE の Score タスクに独自の報酬を追加した Stochastic MuZero を基にしたエージェントの方が，Score タスクの報酬とエージェントのレベルおよびダンジョンレベルに関して優れた性能を発揮した．

本論文の構成は以下のとおりである．第 2 章で関連研究について，第 3 章で本研究の提案手法について，第 4 章で提案手法による実験結果および評価について，第 5 章で考察について，第 6 章で本研究のまとめ及び今後の展望について記す．

2. 関連研究

2.1 The NetHack Learning Environment

Küttler ら[6]は，NetHack Learning Environment (NLE) を提供し，ベースラインモデルの結果を提示した．NLE とは，NetHack 3.6.6 を中心にした Gym 環境の同期型強化学習インターフェースである．NLE では，ゲームのダイナミクスへの変更は加えていない．

NLE の観測空間は，ダンジョンの 2D シンボル観測空間を表す glyphs，chars，colors，specials，エージェント座標とキャラクタ属性を表す blstats，表示されているメッセージを表す message，インベントリアイテムを表す inv_* (*は glyphs，strs，letters，oclasses) から構成される．NLE では，93 の利用可能な行動空間を持つ．また，NLE には，異なる報酬関数と行動空間を持つ 7 つの初期タスクが含まれる．

著者らは，NetHack を機械学習アプローチが攻略することは当面困難であると考えた．ゲーム内スコアはエージェントが達したダンジョンの深さや倒した敵の数などの関数であり，NLE の進歩の代理として賢明であると主張した．そこで，ゲーム内スコアを報酬関数としている score タスクを用いることを推奨している．

ベースラインモデルは，IMPALA と RND を提供している．観測空間は，glyphs と blstats およびエージェントを中心とした 9×9 の範囲を利用した．行動空間は，23 次元に縮小した行動空間を使用した．ベースラインモデルの score タスクにおいて，10 億環境ステップ学習した後の評価の結果は，中立な人間の男性モンクのとき，IMPALA の平均スコアは 748 であり，RND 探索の平均スコアは 780 であり，平均到達ダンジョンレベルは，ともにレベル 5.4 であると報告した．

2.2 ローグライクゲームの深層強化学習研究

MiniHack [3]とは，ユーザが新しい複雑な環境を簡単に作成できるサンドボックス強化学習フレームワークであり，既存のタスク群を多様に公開している．また，MiniHack は NetHack の概念を用いており，強化学習エージェント能力を幅広くテストするベースラインエージェント群を提供している．タスク群の一つであるナビゲーションタスクでは，単純なタスクはすぐに解決できたが，複雑なタスクを追加すると，ベースラインでは進歩しなかった．

金子ら[4]は，ローグライクゲームである Rogue を基にした実験環境 Rogue-Gym を提案し，Double DQN で実際に Rogue-Gym を用いた実験を行った．Rogue-Gym では，敵がおらず，アイテムがゴールドしかなく，階段を下り，ダンジョンの 2 階に降りることができればクリアという単純な環境であった．ダンジョンで拾得したゴールド数を報酬とし，下の階へ降りたときに疑似報酬として 1000 を加えた．その結果，Double DQN の獲得報酬は 1004 であった．したがって，Rogue-Gym では，階段を下りることはできたが，ダンジョンの探索は十分でないという結果が得られた．

NetHack Challenge [7]とは，NeurIPS 2021 Competition and Demonstration Track にて行われたコンペティションである．環境には NLE を使用し，行動空間は 113 行動空間を用いた．コンペティションの結果，ベースラインモデルを上回ったエージェントは複数あったが，いずれのエージェントもゲームをクリアできなかった．

2.3 MuZero

MuZero[1]は，事前知識を全く用いずに，Atari 2600 ゲームにおいて，人間や他の深層強化学習エージェントより優秀なスコアを記録し，囲碁，将棋，チェスにおいて，AlphaZero と同等以上の性能を発揮した．

MuZero は，AlphaZero の探索・方策反復アルゴリズムをベースに，学習モデルを学習手順に組み込んだものであり，シングルエージェント空間や中間時間ステップの非ゼロ報酬など，より幅広い環境へ拡張した．

MuZero のアルゴリズムは，計画に直接関係する未来の側面を予測する．図 2 は，MuZero の探索の方法を表した図である．

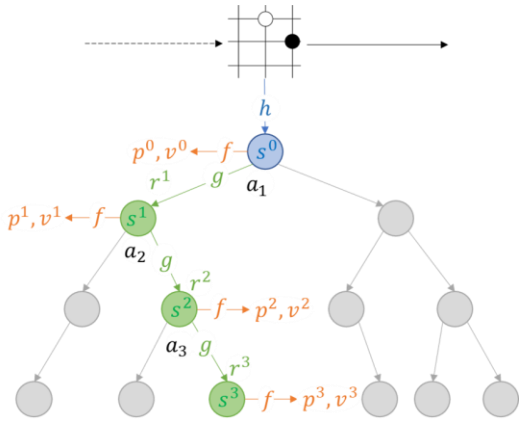

図 2　MuZero の探索の模式図

MuZero のアルゴリズムにおける予測は各タイムステップ t において，$k=0,\cdots,K$ ステップのそれぞれについて，過去の観測値 $o_{\leq t}$ と将来の行動 $a_{t+1},\cdots,a_{t+k}$ $(k>0)$ を条件とするパラメータ θ のモデル μ_θ により行われる．このモデルは，方策 $p_t^k \approx \pi(a_{t+k+1}|\, o,a)$，価値関数 $v_t^k \approx \mathbb{E}[u_{t+k+1}+\gamma u_{t+k+2}+\cdots|\, o,a]$，即時報酬 $r_t^k \approx u_{t+k}$ $(k>0)$ を予測する．なお，u は観測された真の報酬，π は実際の行動を選択するための方策，γ は割引率である．

MuZero は 3 つの関数で表現される．表現関数 h は過去の観測値を符号化して根状態 s_0 を初期化し，予測関数 f は方策 p_k と価値関数 v_t を計算し，ダイナミクス関数 g は仮想ステップ k ごとに即時報酬 r^k と内部状態 s^k を計算する．なお，ダイナミクス関数が決定論的であるため，確率的遷移をする環境に対応していない．

MuZero の MCTS アルゴリズムはシングルエージェント領域と中間報酬を許容するように一般化された．各内部ノードにおいて，探索木の根で改良方策 π_t と改良価値関数 v_t を生成する．次の行動 $a_{t+1} \approx \pi_t$ は pUCT アルゴリズムに従って探索方針から選択する．

モデルのすべてのパラメータは，仮想ステップ k ごとに，方策，価値関数，報酬予測を，実際のタイムステップが k 回経過した後に観測された 3 つの対応するターゲットに一致するように同時に学習される．MuZero の損失関数は，式 3 のようである．式 3 の損失関数では，方策 p_t^k と探索方策 π_{t+k} の誤差と，価値関数 v_t^k と価値目標 z_{t+k} の誤差，および予測即時報酬 r_t^k と観測即時報酬 u_{t+k} の誤差を最小化することが目的であり，右辺の最終項最後は L2 正則化項である．

$$L_t^{MuZero}(\theta) = \sum_{k=0}^{K} l^p\left(\pi_{t+k}, p_t^k\right) + \sum_{k=0}^{K} l^v\left(z_{t+k}, v_t^k\right) + \sum_{k=1}^{K} l^r\left(u_{t+k}, r_t^k\right) + c\|\theta\|^2 \tag{3}$$

2.4　Stochastic MuZero

Stochastic MuZero [8]は，前節に挙げた MuZero の確率的な環境に適応できないという欠点を解決した．Stochastic MuZero は，2048 とバックギャモンにおいて MuZero より優れた報酬を得ることができ，囲碁において MuZero と同等の性能を発揮した．

Stochastic MuZero は，確率的ダイナミクスを捉えるため，事後状態 as_t の概念を利用した．Stochastic MuZero では，MuZero に事後状態を導入し，環境に行動を適用したときの効果と，行動が与えられたときの偶然の遷移の効果を分離した．Stochastic MuZero では，環境が遷移する可能性のある状態 M は有限であると仮定し，各遷移を偶然の結果 c_t^i と関連付けた．そして，確率モデルの学習は事後状態 as と偶然の結果 c を学習する問題に帰着した．

Stochastic MuZero の確率モデルは，5 つの関数から構成される．表現関数 h は現在の観測値 $o_{\leq t}$ を隠れ状態 s_k^t に写像する．予測関数 f は状態 s_k^t が与えられると価値 v_k^t と方策 p_k^t の予測を生成する．事後状態ダイナミクス関数 ϕ は状態 s_t^k と行動 a_{t+k} が与えられたときに次の潜在的事後状態 as_k^t を生成する．事後状態予測関数 ψ は事後状態 s_t^k が与えられると，値予測 Q_t^k と，将来の可能な偶然の結果 c_{t+k+1} に対する分布 $\sigma_t^k = \Pr(c_{t+k+1}|\, as_k^t)$ を生成する．ダイナミクス関数 g は事後状態 as_k^t と偶然の結果 c_{t+k+1} が与えられたときに次の隠れ状態 s_t^{k+1} と報酬予測 r_t^{k+1} を生成する．

Stochastic MuZero の偶然の結果は，VQ-VAE [10]の変形を用いてモデル化された．サイズ M の定数コードブックを考え，エンコーダの埋め込み c_t^e をカテゴリ関数としてモデル化し，最も近いコード c_t を選択する．なお，VQ-VAE のデコーダ部は利用されなかった．

Stochastic MuZero の確率モデルでは，式 5 のように MuZero ロスとチャンスロスの和を最適化する学習を行なわれる．観測値 $o_{\leq t:t+K}$，行動 $a_{t:t+K}$，価値目標 $z_{t:t+K}$，実報酬 $u_{t+1:t+K}$ を持つ長さ K の軌道が与えられるとき，K ステップ分モデルを展開し，学習される．

$$L^{all} = L^{MuZero} + L^{chance} \tag{5}$$

L^{MuZero} は MuZero の損失関数である式 3 と同一のものである．L^{chance} は事後状態 as_k に対する予測値 Q_t^k および σ_t^k に対して適応される．Q_t^k は価値目標 z_{t+k} に一致するように，σ_t^k はエンコーダが生成するコード c_{t+k+1} に向かって学習される．最後に，標準的な VQ-VAE の実装に従い，VQ-VAE コミットメントコストを使用した．式 6 はチャンスロスを表した式である．

$$L^{chance} = \sum_{k=0}^{K-1} l^{Q}\left(z_{t+k}, Q_t^k\right) + \sum_{k=0}^{K-1} l^{\sigma}\left(c_{t+k+1}, \sigma_t^k\right) + \beta \sum_{k=0}^{K-1} \left\| c_{t+k+1} - c_{t+k+1}^{e} \right\|^2 \quad (6)$$

Stochastic MuZero の探索は，MuZero の MCTS にチャンスノードとチャンス値を導入したものであり，決定ノード（図 3 丸形ノード）とチャンスノード（図 3 菱形ノード）の 2 種類から構成された．チャンスノードでは，そのノードの値と将来のコードに関する事前分布を返す．チャンスノードの展開後，その値は木の上にバックプロパゲートされ，事前分布をサンプリングして確率的にコード c が選択される．また，学習済みモデルを利用して内部決定ノードが再拡張される．決定ノードでは，報酬，価値，方策を返す．新しく追加されたノードは木をバックプロパゲートし，pUCT アルゴリズムを用いて行動が選択される．図 3 は Stochastic MuZero の確率的探索の模式図である．

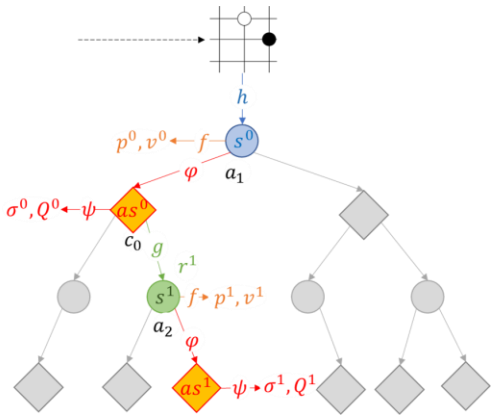

図 3　Stochastic MuZero の探索の模式図

3. 提案手法

本研究では，確率的な環境に対応していないエージェントと対応しているエージェントのどちらが NLE において良いか比較する．そこで，確率的な環境に対応していない MuZero を基にした提案手法 1 と，独自報酬を加えた確率的な環境に対応している Stochastic MuZero を基にした提案手法 2 の二つを提案する．

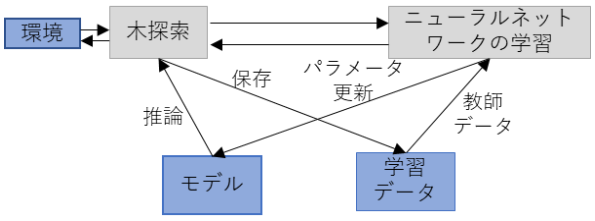

図 4　ネットワークの構成図

それぞれの手法は，MuZero および Stochastic MuZero と同様に，図 4 のように木探索，学習という流れを 1 サイクルとし，これを繰り返すことで，ネットワークを学習させ，ゲームを攻略するエージェントを作成することを目指した．

3.1　モデルへの入力

提案手法のモデルへの入力は，NLE で提供している glyphs，blstats (strength_percentage を除く) および inv_glyphs を組み合わせて 2 次元配列とし，それを現在の局面と現在から 7 つ前までの局面の計 8 つの局面をニューラルネットワークへの入力とした．

3.2　提案手法 1

提案手法 1 の MCTS は MuZero と同様に設計した．また，提案手法 1 のネットワークは 3.2.1 で，ニューラルネットワークの学習は 3.2.2 で説明する．

3.2.1　モデル

提案手法 1 のモデルは MuZero と同様に，表現モデル，予測モデル，ダイナミクスモデルの 3 つのモデルから構成した．なお，3 つのモデルはそれぞれ表現関数 h，予測関数 f，ダイナミクス関数 g に対応する．

3.2.2　提案手法 1 の学習

提案手法 1 では，MuZero のネットワークの学習と同様にエピソードが終了するごとに 3 つのモデルを同時に 16 回学習させた．

学習データから軌跡をサンプリングした．最初に，表現関数 h は選択された軌道から過去の観測値 $o_1, \cdots, o_t$ を入力として受け取り，その後 K ステップにわたって再帰的に展開した．各ステップ k において，ダイナミクス関数 g は前ステップの隠れ状態 s^{k-1} と実際の行動 a_{t+k} を受け取った．表現関数，予測関数，ダイナミクス関数のパラメータは，バックプロパゲーションにより終端まで学習され，方策 $p \approx \pi$，価値関数 $v \approx z$，報酬 $r \approx u$ の 3 つの量を予測した．なお，π，z，u は学習データからランダムサンプリングされたものである．図 6 は提案手法 1 の学習の流れである．

提案手法 1 の損失関数は，MuZero の損失関数を参考に式 7 のようにした．なお，optimizer には Adam を，方策，価値，報酬に関するそれぞれの損失関数 l^p，l^v，l^r には MSELoss を使用した．また，$K = 5$，バッチサイズは 64 とした．

$$L(\theta) = \sum_{k=0}^{K} l^{p}\left(\pi_{t+k}, p_t^k\right) + \sum_{k=0}^{K} l^{v}\left(z_{t+k}, v_t^k\right) + \sum_{k=1}^{K} l^{r}\left(u_{t+k}, r_t^k\right) \quad (7)$$

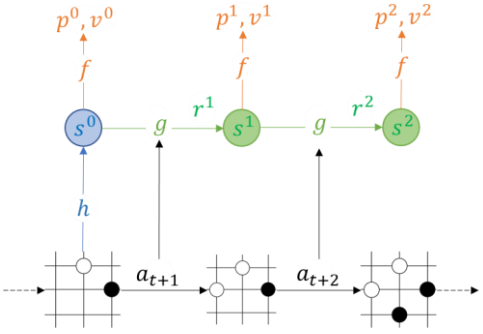

図 6　提案手法 1 の学習方法の模式図

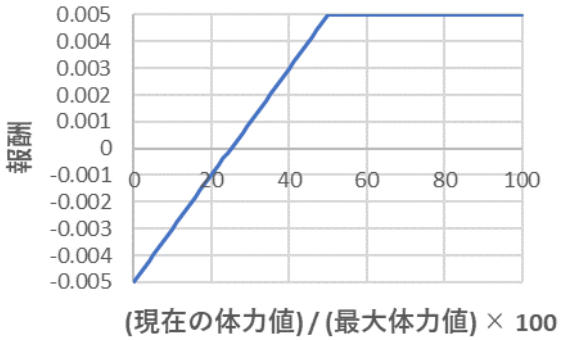

図 7　体力の割合により与える報酬

3.3　提案手法 2

提案手法 2 では，NLE の Score タスクに報酬を追加した．そこで，その追加報酬について 3.3.1 で説明する．また，提案手法 2 の MCTS は 3 手先までを先読みするように制限したが，他は Stochastic MuZero と同様に設計した．提案手法 2 と Stochastic MuZero で異なるモデルは 3.3.2 で，ニューラルネットワークの学習は 3.3.3 で説明する．

3.3.1　追加報酬

提案手法 2 では，次の 6 点を追加報酬として加えた．

1. 行動空間の 0 番目の行動をとったとき -0.5
2. ダンジョンの階段を下りると
   $+200 \times$ (降りた先のダンジョンレベル)
3. 2 以外でダンジョンの探索が進む（glyphs=2359 が減る）と $+0.01 \times$ (glyphs=2359 の減少分)
4. エージェントのレベルが上がると $+50$
5. 空腹度が not hungry (blstats[21] $= 1$)のとき $+0.05$,
   satiated (blstats[21] $= 0$)のとき -0.02,
   その他のとき $(1 - \text{blstats}[21]) * 0.1$ (blstats[21] ≥ 2)
6. それぞれのステップごとに
   $\min[(\text{現在の体力})/(\text{最大体力}) - 0.25, 0.25]/50$

1 では，0 番目の行動は何もしないため，ペナルティとして -0.5 の報酬を与えた．2 では，ダンジョンの攻略と階段を下りることの関連性が強いことから，ダンジョンを下りると，ダンジョンレベルに比例する大きな正の報酬を与えた．3 では，同一階層で探索を進めることを意図し，未探索部である glyphs = 2359 が減少した分だけ正の報酬を与えた．4 では，エージェントの成長を意図し，レベルが上がるたびに一定の正の報酬を与えた．5 では，空腹度が空腹でないのときのみ正の報酬を与え，空腹度が満たされているときと，空腹であるときは負の報酬を与えた．6 では，エージェントの体力に応じて図 7 のように報酬を与えた．

3.3.2　提案手法 2 のモデル

提案手法 2 のモデルは Stochastic MuZero と同様に，表現モデル，予測モデル，事後状態ダイナミクスモデル，事後状態予測モデル，ダイナミクスモデルの 5 つのモデルから構成した．なお，各モデルはそれぞれ表現関数 h，予測関数 f，事後状態ダイナミクス関数 ϕ，事後状態予測関数 ψ，ダイナミクス関数 g に対応する．

3.3.3　提案手法 2 の学習

提案手法 2 の学習では，Stochastic MuZero の学習と同様に 5 つのモデルをエピソードが終了するたびに学習を同時に 16 回実行した．

エンコーダ e は観測値 $o_{\leq t+k}$ を入力として受け取り，チャンスコード c_{t+k} を決定論的に生成した．モデルの政策 p_{t+k}，価値 v_{t+k}，即時報酬 r_{t+k}，予測値 Q_{t+k} はそれぞれ目標値 π_{t+k}，z_{t+k}，u_{t+k}，z_{t+k} に向かって学習された．また，事後状態 as に対する σ_{t+k} はエンコーダの出力 c^e_{t+k} に，エンコーダの出力 c^e_{t+k} はコード c_{t+k} に向かって学習された．なお，それぞれの目標値は学習データからランダムサンプリングされた値であった．図 8 は提案手法 2 の学習の流れである．

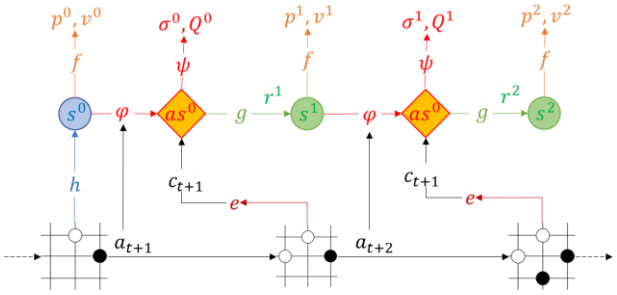

図 8　提案手法 2 の学習の流れの模式図

提案手法 2 の損失関数は，Stochastic MuZero の損失関数を参考に式 8 のようにした．なお，optimizer には Adam を，方策に関する損失関数 l^p，価値に関する損失関数 l^v，報酬に関する損失関数 l^r，予測値 Q^k_t と σ^k_t に関する損失関数 l^Q および l^σ，VQ-VAE に関する損失関数

l^c には MSELoss を使用した．また，K を 5，バッチサイズを 64 とした．

$$L^s(\theta) = L'(\theta) + L^c(\theta)$$

$$L'(\theta) = 100\sum_{k=0}^{K} l^p\left(\pi_{t+k}, p_t^k\right) + 0.01\sum_{k=0}^{K} l^v\left(z_{t+k}, v_t^k\right) + 0.05\sum_{k=1}^{K} l^r\left(u_{t+k}, r_t^k\right) \quad (8)$$

$$L^c(\theta) = 0.01\sum_{k=0}^{K-1} l^Q\left(z_{t+k}, Q_t^k\right) + 20\sum_{k=0}^{K-1} l^\sigma\left(c_{t+k+1}, \sigma_t^k\right) + 20\sum_{k=0}^{K-1} l^c\left(c_{t+k+1}, c_{t+k+1}^e\right)$$

3.4　行動選択

各提案手法の行動選択確率 $\Pr(a)$ を求める際，式 9 のように温度パラメータ τ を使用した．$N(a)^{1/\tau}$ は各行動の選択回数，$\sum_b N(b)^{1/\tau}$ は 1 行動あたりのシミュレーション回数を示す．

$$\Pr(a) = \frac{N(a)^{1/\tau}}{\sum_b N(b)^{1/\tau}} \quad (9)$$

4. 実験・評価

提案手法 1 と提案手法 2 についてそれぞれ実験を行った．それぞれの実験を実験 1，実験 2 とする．

各実験では，4000 エピソードの学習を行い，その直後に 100 エピソードの評価を行った．なお，学習において，最初の 14 エピソードはデータを蓄積するために学習は実行しなかった．また，各実験において，エピソード数は 0 から数えることにした．

実験 1，実験 2 では，それぞれ 1 手当たりのシミュレーション回数は 12 回，割引率 γ は $\gamma = 0.99$ とした．また，MCTS の根における exploration noise のパラメータおよび，MCTS の行動選択に使用する pUCT の UCB 計算のパラメータは MuZero と同一の値とし，評価時の温度パラメータは 0.1 とした．

なお，両実験・評価とも，NetHack のロールはモンク，種族は人間，性別は男性，立場は中立で行った．提案手法 1 の実験である実験 1 は 4.1 で，提案手法 2 の実験である実験 2 は 4.2 で述べる．

4.1　実験 1

学習率は 0.005 にした．また，温度パラメータ τ は表 1 のように，エピソードが経過するとともに減少するようにした．

表 1　実験 1（提案手法 1）の温度パラメータ

エピソード数 epi	温度パラメータ τ
$14 \leq epi < 500$	1
$500 \leq epi < 1000$	0.75
$1000 \leq epi < 1500$	0.65
$1500 \leq epi < 2000$	0.55
$2000 \leq epi < 3000$	0.3
$3000 \leq epi < 4000$	0.25

4.1.1　実験 1 の学習結果

実験 1 の獲得報酬の推移すなわち学習曲線の区間 100 の移動平均は図 9 のようになった．また，学習中に達したダンジョンレベルとエージェントのレベルの回数は表 2 のようになった．なお，学習におけるステップ数は 672 万環境ステップであった．

図 9　実験 1 の学習曲線

表 2　提案手法 1 の学習時におけるエージェントのレベルとダンジョンレベル

達したレベル	ダンジョンレベル（回）	エージェントのレベル（回）
1	3458	3871
2	461	125
3	62	4
4	16	0
5	3	0

4.1.2　提案手法 1 の評価

提案手法 1 では，4000 エピソードの学習を実行した後，100 エピソードの評価を行った．その結果，エージェントが獲得した報酬の平均値と最大値は表 3，ダンジョンレベルおよびエージェントのレベルは表 4 のようになった．

表 3　提案手法 1 の報酬の評価

最大報酬	平均報酬
171.73	19.40

表 4　提案手法 1 のダンジョンレベルと
エージェントのレベルの評価

達したレベル	ダンジョンレベル（回）	エージェントのレベル（回）
1	88	99
2	11	0
3	1	1
平均	1.13	1.02

4.2　実験 2

VQ-VAE 部のコードブックサイズ M は，$M = 32$ とし，学習率および温度パラメータ τ は表 5 のようにした．

表 5　実験 2 の学習率と温度パラメータ

エピソード数 epi	学習率	温度パラメータ τ
$15 \leq epi < 500$	0.01	1
$500 \leq epi < 1000$	0.008	0.8
$1000 \leq epi < 1500$	0.006	0.5
$1500 \leq epi < 2000$	0.005	0.4
$2000 \leq epi < 3000$	0.003	0.3
$3000 \leq epi < 4000$	0.001	0.25

4.2.1　実験 2 の学習結果

実験 2 における Score タスクの報酬と独自の報酬の和である総報酬つまり，学習曲線は図 10，Score タスクのみの報酬は図 11 のようなった．また，学習中のダンジョンレベルとエージェントのレベルの達した回数は表 6 のようになった．なお，学習におけるステップ数は 662 万環境ステップであった．また，図 10 および図 11 は区間 100 の移動平均である．

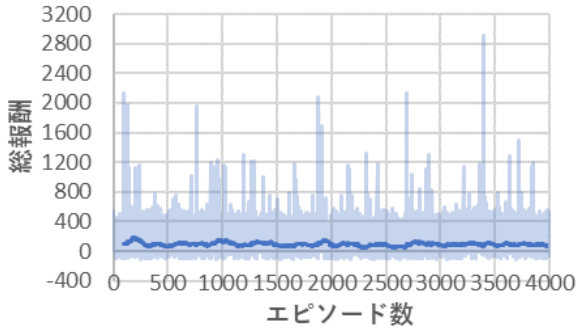

図 10　実験 2 の学習曲線

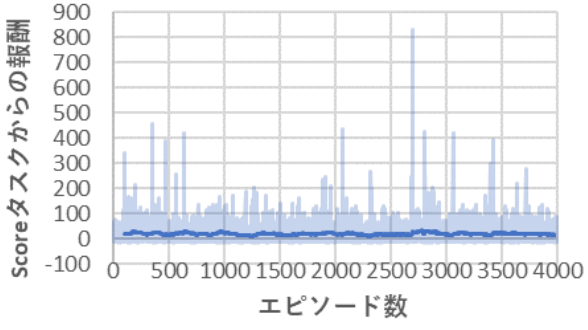

図 11　実験 2 の Score タスクのみの報酬

表 6　提案手法 2 の学習時におけるエージェントのレベルとダンジョンレベル

達したレベル	ダンジョンレベル（回）		エージェントのレベル（回）	
	実験 1	実験 2	実験 1	実験 2
1	3496	3458	3894	3871
2	443	461	100	125
3	48	62	6	4
4	6	16	0	0
5	5	3	0	0
6	1	0	0	0
7	1	0	0	0

4.2.2　提案手法 2 の評価

提案手法 2 について，4000 エピソードの学習を実行した後，100 エピソードの評価を行った．その結果，エージェントが獲得した報酬の平均値と最大値は表 7，ダンジョンレベルおよびエージェントのレベルは表 8 のようになった．なお，表 7 の Score は，Score タスクから与えられた報酬のみの値を表す．

表 7　提案手法 2 の報酬の評価

最大報酬	平均報酬	Score 最大	Score 平均
1207.34	125.85	199.54	24.08

表 8　提案手法 1 のダンジョンレベルと
エージェントのレベルの評価

達したレベル	ダンジョンレベル（回）	エージェントのレベル（回）
1	83	98
2	13	1
3	3	1
4	0	0
5	1	0
平均	1.23	1.03

5. 考察

表 3 より，評価における提案手法 1 の平均報酬は 19.40，最大報酬は 171.73 であった．一方で，表 7 より，提案手法 2 の Score タスクから与えられた報酬は，平均報酬が 24.08，最大報酬が 199.54 であった．よって，報酬では提案手法 2 の方が優れていることが分かった．また，表 4 より，提案手法 1 の平均ダンジョンレベルは 1.13，エージェントの平均レベルは 1.02 であった．一方で，表 8 より，提案手法 2 の平均ダンジョンレベルは 1.23，エージェントの平均レベルは 1.03 であった．よって，エージェントのレベルおよびダンジョンレベルにおいても提案手法 2 の方が優れていた．したがって，確率的な環境に対応している提案手法 2 の方が良い手法であると示された．

提案手法 2 の方が良くなった要因は二つあると考えられる．一つ目の要因は，NetHack とそれを基に作成

された NLE が確率的な環境を持つことである. 提案手法 1 は, MuZero を基に作成しているため, 確率的な環境に対応していない. 一方で, 提案手法 2 は, Stochastic MuZero を基に作成しているため, 確率的な環境に対応している. したがって, 確率的な環境である NLE では, 確率的な環境に対応している提案手法 2 の方が良い結果になったと考えられる. 二つ目の要因は, 追加報酬の働きである. 提案手法 2 では, 攻略の助けになるような報酬を追加した. その結果, Score タスクからの報酬だけを見ても提案手法 2 の方がより多くの報酬を獲得した可能性がある.

一方で, 図 9, 10, 11 より, 実験 1, 2 の学習曲線と, 実験 2 の Score タスクの報酬は増加が見られない. また, 評価において, 表 7, 8 より, 提案手法 2 の Score タスクから与えられた報酬の平均値は 24.08, 平均ダンジョンレベルは 1.23 であり, 2.1 より, NLE のベースラインモデルの IMPALA の平均スコアは 748, 平均ダンジョンレベルは 5.4 であったため, 提案手法 2 はベースラインモデルには及ばなかった.

報酬が横ばいで推移し, 評価において NLE のベースラインモデルの結果を下回った原因は三つあると考えられる. 一つ目は, シミュレーション回数が 12 回と少なかったことである. [8]では, 囲碁のシミュレーション回数は 800 回のため, 本手法では[8]の囲碁の 1.5 % のシミュレーション回数であった. したがって, シミュレーション回数が不足していたのではないか. 二つ目は, 学習回数が少なかったことである. 一般に, 複雑な環境下の深層強化学習では, 学習を多く行うことが必要である. しかし, 本研究では, $16 \times (4000 - 14) = 63776$ 回に過ぎない学習回数であった. よって, 複雑な環境下において十分な学習が行われなかったと考えられる. 最後に三つ目は, 提案手法 1, 2 において, モデルを小さくして実験した. そのため, 確率性や複雑性を持つ環境において, エージェントが良い行動を十分に発見できなかった可能性がある.

6. まとめ

本研究では, NLE の Score タスクを環境として利用して, 確率的な環境に対応していない MuZero を基にした提案手法 1 と, 確率的な環境に対応している Stochastic MuZero を基にし, 独自報酬を追加した提案手法 2 について, それぞれ 4000 エピソードの学習と 100 エピソードの評価を行った. 評価の結果, 提案手法 1 より提案手法 2 の方が良い成績であった. したがって, NLE を環境とした場合, 確率的な環境に対応していない提案手法 1 より, 確率的な環境に対応している提案手法 2 の方が良いエージェントであることが示された.

今後の課題は三つある. 一つ目は, シミュレーション回数や学習回数を多くすることである. 先述のように, 本研究のシミュレーション回数はわずか 12 回であった. また, エピソード数が少なかった. これらが要因となり, 十分に学習できなかった可能性がある. よって, シミュレーション回数やエピソード数を大きくすることを検討したい. 二つ目は, 損失関数などの学習設定の再検討である. 今回, 損失関数はスケーリングを行わなかったため, スケーリングすることを検討したい. また, 今回, エピソードの終了時に学習を行った. しかし, NLE は最大 5000 ステップと長いため, 一定のステップ数で区切って学習させることを検討したい. 三つ目は, 提案手法 2 の追加報酬の見直しである. 今回の追加報酬が適切であったか, 追加報酬の項目や与える報酬の値を検討したい.

謝　辞

本研究は JSPS 科研費 JP21H03496, JP22K12157 の助成を受けたものです. また, 本研究は, 電気通信大学人工知能先端研究センター(AIX)の計算機を利用して実施したものです.

参　考　文　献

[1] Julian Schrittwieser, Ioannis Antonoglou, et al, “Mastering Atari, Go, Chess and Shogi by planning with a learned model”, Nature Vol. 588. 604-609, 2020.

[2] Oriol Vinyals, Timo Ewalds, et al., “StarCraft II: A New Challenge for Reinforcement Learning”, arXiv: 1708.04782, 2017.

[3] Mikayel Samvelyan, Robert Kirk, et al., ”MiniHack the Planet: A Sandbox for Open- Ended Reinforcement Learning Research”. 35th Conference on Neural Information Processing Systems, 2021.

[4] 金川裕司, 金子知適, “ローグライクゲームによる強化学習ベンチマーク環境 Rogue-Gym の提案”, 情報処理学会　第 23 回ゲームプログラミングワークショップ, 2018.

[5] NetHack 3.6.6. https://www.nethack.org/

[6] Heinrich Küttler, Nantas Nardelli, et al., “The NetHack Learning Environment”, 34th Conference on Neural Information Processing Systems, 2020.

[7] Eric Hambro, Sharada Mohanty, et al., “Insights from the NeurIPS 2021 NetHack Challenge”, Proceedings of Machine Learning Research 176:41–52, 2022.

[8] Ioannis Antonoglou, Julian Schrittwieser, et al., “Planning in Stochastic Environments with a Learned Model”, Tenth International Conference on Learning Representations, 2022.

[9] NetHack wiki. https://nethackwiki.com/wiki/Main_Page

[10] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu “Neural Discrete Representation Learning”, 31st Conference on Neural Information Processing Systems, 2017.