アクションゲームにおけるディープニューラルネットワーク付きステートマシンを用いたキャラクターAI強化学習

周 済涛[1,a)]　三宅 陽一郎[1,b)]

概要：強化学習を用いたエージェント設計が進歩を遂げており，より効率的かつ柔軟に強化学習をコントロールする手法が求められている．そこで，古典的な意思決定モデルであるステートマシンとディープニューラルネットワーク（DNN) 強化学習の組み合わせを検証する．各ステートは一つの DNN を持ち学習を実行する．学習中，遷移によってステートの切り替えが発生した場合，ノンアクティブになったステートに対応する DNN の学習は一旦停止されるが，再びアクティブになった場合に学習が再開される．アクティブになったステートの DNN がキャラクターを制御する．DNN 付きステートは，記号主義的に定義されたステートとコネクショニズムのニューラルネットがセットになっており，より柔軟な制御を持つキャラクター AI を作成できる．本研究は，Unity3D の環境でステートマシンを構築し，強化学習をコントロールしながらキャラクター AI にゲーム内で戦闘する方法を学習させた．多数の DNN を並行して学習させたモデルを 1 つの DNN で学習させたモデルを比較し，本手法全体の性能を評価する．

Character AI Reinforcement Learning by Using Finite State Machine with Deep Neural Network in Action Games

Jitao Zhou[1,a)]　Youichiro Miyake[1,b)]

Abstract: Agent design by using reinforcement learning has been making progress, and there is a requirement for more efficient and flexible methods to control reinforcement learning. We examine the combination of a classical decision-making model, the state machine, and deep neural network (DNN) reinforcement learning. Each state has a DNN and performs learning. During learning, when a state switch occurs due to a transition, the learning of the DNN corresponding to the inactive state is stopped, but resumes when the state becomes active again. The DNN of the activated state controls the character, and the DNN state consists of symbolically defined states and connectionism neural nets. It allows the creation of character AI with more flexible control. In this study, a state machine is built in the Unity3D environment and a character AI is trained to learn how to fight in a game while the state machine controls reinforcement learning. We evaluate the overall performance of our method by comparing two models: the one is trained with many DNNs, and the other is trained with a single DNN.

1. はじめに

強化学習とは，エージェントの人工知能の構築に用いられている環境とインタラクションしながら学習する機械学習の一種である．近年，ディープニューラルネットワークの発展に伴い，深層強化学習という手法が生み出され，様々な分野で大きな成果をあげている．例えば，DeepMind 社の Agent57 が Atari57 全てのゲームで人間を超えるレベルに到達している [1].

しかし，ゲームプレイで人を超える能力を発揮するディープニューラルネットワークが，ゲーム産業のリリースされたゲームでキャラクター AI 開発に使用された例は非常に少ない．その理由は，キャラクター AI は，強さ，賢さを求められながら，キャラクターの個性を出す必要があると

[1] 立教大学大学院　人工知能科学研究科
Graduate School of Artificial Intelligence and Science, Rikkyo University
[a)] 21vr014n@rikkyo.ac.jp
[b)] youichiro_miyake@rikkyo.ac.jp

ころにある．デジタルゲームにおけるゲーム AI は強いだけではなく，ゲーム開発者の想定通りの動きを表現する必要がある．一定のパターンを持つ，キャラクターの個性を出すのがゲーム的な AI である．ニューラルネットワークの多数のパラメータの意味を解釈することは難しく，パラメータからニューラルネットワークを調整しゲーム AI の行動パターンを調整するのは不可能である．そこで強化学習を用いてある程度意図したように学習させる方法が，強化学習を用いたキャラクター AI 作成の鍵となる．

そこで，古典的な意思決定モデルとコネクショニズムのディープニューラルネットワークを組み合わせた手法が考えられる．有限ステートマシン (FSM) は 90 年代から 00 年代初頭まで，ゲーム産業においてキャラクター AI を作る最も人気の高かった手法である．各ステートに明確な役割を与えることで，各ステート内での動きは明確になる．本研究では，ステートマシンと強化学習の組み合わせについて，ディープニューラルネットワーク付きのステートマシンモデルを提案する．DNN をステートマシンで分割し，分割したそれぞれの DNN に対して意図した一定のパターンを学習させる．実行時には，ディープニューラルネットワークをステートマシンの切り替えによって動作させ，キャラクターを制御する．実験では，Unity3D で本手法を用いたキャラクター AI を作成し，ステートマシンを使わない従来の強化学習の手法と比較して評価する．

2. 研究背景

2.1 有限ステートマシン

有限ステートマシン (FSM) は 90 年代からゲーム産業でキャラクター AI 作成の手法として,「Quake」や「UNCHARTED2」など数千以上のタイトルで長年使われている [2]．一般的な FSM は $(\Sigma, S, s_0, \delta, F)$ という五つの要素から構成されている．Σ は入力の集合，S は状態の集合であり，δ:S $\times \Sigma \rightarrow$ S は状態遷移関数である．s_0 と F はそれぞれ開始状態と終了状態の集合であり，S の部分集合でもある．ゲームにおける FSM は，状態，アクション，遷移条件で構成され，それぞれの状態でキャラクターは設定されたアクションをとり，遷移条件を満たすと該当の状態への遷移が発生する．近年ゲームの複雑化に伴い，単純なステートマシンの代わりに，階層型ステートマシン (HFSM) を用いて，キャラクター AI を作成する事例が増えている [3].

2.2 深層強化学習

ニューラルネットワークは人間の脳を模倣した数理モデルであり，1943 年から提唱され，近年多層化した階層構造を持つとなり，大きな発展を遂げている．ニューラルネットワークは入力層，隠れ層，出力層から構成される．学習する場合，入力層からデータを入力し，正解な出力が出るように隠れ層で大量の重みを更新し学習する．推論する場合，入力層からデータを入力し，隠れ層で重みと掛け算など複雑な計算を行い，最後に出力層で推論を出力する．一方，強化学習とは，現在の状態を観測し，過去からの経験から学習し，報酬を最大化するための学習手法である．強化学習をモデル化すると，以下の要素から構成する．状態 $s \in S$，座標などエージェントがいる状況を表す．行動 $a \in A$，今の状態で取れる行動である．方策 $\pi : S \rightarrow A$，状態において取るべき行動を表す．報酬 $r(s, a, s')$，状態 s のエージェントが行動 a を取り，状態 s' に遷移して得られる報酬である．方策 π を固定し，状態 s で行動 a を取った場合，将来が得られる報酬の合計が行動価値関数 $Q^{\pi}(s, a)$ と呼ばれる．強化学習は主に価値ベースと方策ベース両方が分かれている．価値ベースのアルゴリズムは，価値関数 Q の最大値 $Q^{*}(s, a)$ に対して，どんな状況でどんな行動を取るかを遡って学習する．方策ベースは報酬を最大するため，直接方策 π を学習する．ディープニューラルネットワークと強化学習を組み合わせた結果が深層強化学習である．代表的な価値ベースの深層強化学習アルゴリズムは Q 学習をディープ化した DQN である [4]．代表的な方策ベースの深層強化学習アルゴリズムは PPO[5]，SAC[6] などがある．

3. 関連研究

Pierre-Luc Bacon らは，大きな Policy を使い，Option の使い分けをする研究を行った [7]．学習途中大きな Policy と Option 中の Policy と同時に更新される．しかしステートの概念は含まれてない，Policy の使い分けも人に関与できないである．

Patrick Klose らは，自動生成のステートで強化学習を分割し，自動運転に適用するの研究を行った [8]．ステート分割によって，行動空間が制限され，DQN を用いた強化学習が必要な計算量は減少し，学習を加速度させることができる．しかし，自動生成のステートは思い通りにコントロールできないため，ゲーム産業に適してないと考えられる．

Li らは，強化学習とビヘイビアツリーの組み合わせについて研究を行った [9]．ビヘイビアツリーはステートマシンと同様にゲーム産業で AI 作成に使われている手法である．これらの研究で，強化学習ノードと普通の記号主義で定義されたノードは矛盾なく存在することが証明される．

上段らは，強化学習を通じて敵キャラクター AI をバトルの仕方を学習し，学習後 AI に調整を加えてゲームキャラクターの要求に沿った AI を作成した [10]．プランナーが学習の方向性を決め，学習後の AI に対して報酬の修正によって，特定なアクションが出る確率を高めた．しかし，ニューラルネットワークは 1 つしか使用していないため，多環境に対応するには難しいという問題もあった．以上の先行研究を踏まえて，ディープニューラルネットワーク付

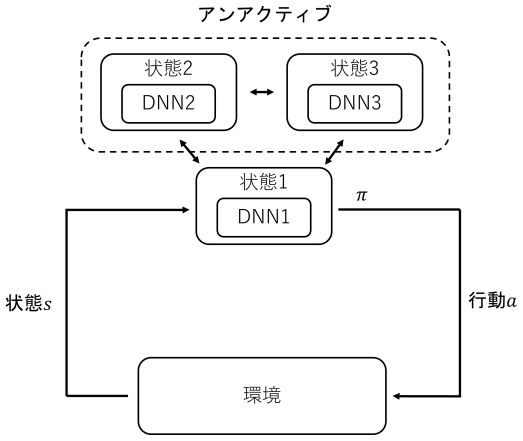


図 1 アーキテクチャ

きのステートマシンという新しい手法を提案する．

4. 提案手法

本手法は一般的なステートマシンの各ステートにディープニューラルネットワークを付与したアーキテクチャとなる (図 1)．つまりステート内部のアクション部分にニューラルネットワークを埋め込んだアーキテクチャである．一般的なステートと同じように，ステート間に遷移条件があり，遷移条件が満足した場合，ステート間の遷移が発生する．遷移と同時に発生するディープニューラルネットワークの切り替えによって，エージェントの動きのパターンは変化する．

学習時，ステートの切り替えが発生する場合，現在の DNN の学習を一旦停止させ，次のステートに付いている DNN に切り替え，学習を開始させる．現在アクティブとなった DNN だけが更新される．このように，多数の DNN モデルが切り替えしながら学習を並行して行う．

本手法の利点は，ステートマシンは開発者が自由にカスタマイズすることができ，逆にステート内は制限された状態空間の中で DNN が自律的に学習する，という点にある．これによって，開発者から見た場合，ステートをする定義だけで，あとは DNN が自動的にそのステート内部の学習を行ってくれることになる．

5. 実験

5.1 実験概要

本研究は Unity3D 上で実装と検証を行う．強化学習は The Unity Machine Learning Agents Toolkit(ML-Agents)[11] を使用している．ML-Agents はオープンソースプロジェクトであり，Python API を通じて，強化学習エージェントの訓練をより効率的に行うことを可能にする．

実験環境は図 2 で表す．20*20 のフィールドでいくつかの障害物を配置し，一体のステートマシンで制御する敵

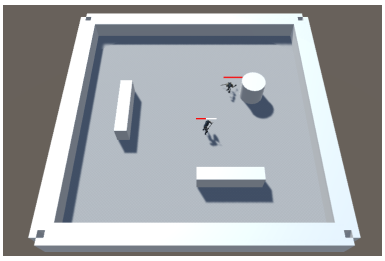
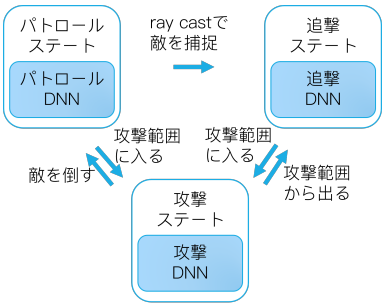


図 2 実験環境



状態遷移図

図 3 状態遷移図

キャラクター (強化学習を使わない) を対象として，本手法を用いて剣で攻撃するキャラクター (エージェント) にバトルの仕方を学習させる．

エージェントはパトロール，追撃，攻撃 3 つの状態 (ステート) を持ち，各状態で行える行動に制限をかけている．パトロールと追撃状態は，前後左右の移動と回転の行動しか出せない，攻撃状態は上記の移動に攻撃と防御行動を加えて行えるようになる．エージェントの状態遷移は図 3 で示されたように，前方にに発射する光線が敵に当たると追撃状態に移行し，敵がエージェントの攻撃範囲に入ると攻撃状態に移行する．敵を倒したらまたパトロール状態に戻る．

観測値として，エージェントと敵の相対位置座標，敵のステート情報と敵は攻撃行動を行っているかどうか，そしてキャラクター正面から発射する 3 本の光線が敵に当たっているかどうか，合計 14 個の情報をエージェントに与える．

敵キャラクターは同じくパトロール，追撃，攻撃 3 つの状態ステートを持つ．敵キャラクターは一定の間隔でフィールド上のランダムな位置に移動し，エージェントが敵キャラクターの感知範囲内に入る場合，エージェントの位置まで移動し，攻撃範囲に入ると攻撃するというように設定されている．

5.2 比較実験

比較実験として，本手法で学習させたエージェントと，他の二種類の手法で学習させたエージェントを比較する．

比較対象の一つ目はステート情報を入力として，1 つの DNN で学習させたエージェントである．ステートの切り

表 1　ステートごとの報酬設置

ステート	プラス報酬	マイナス報酬
パトロール	ray cast で敵を捕捉：+1	行動をするたびに：-0.001 敵の攻撃が自分に当たる：-1
追撃	敵が自身の攻撃範囲に入る：+1	
攻撃	攻撃が敵に当たる：+1	

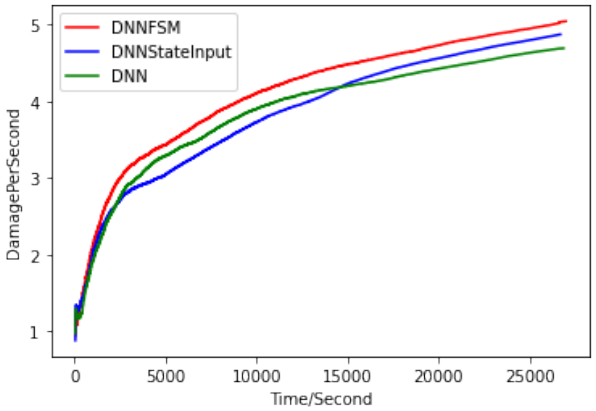


図 4　キャラクターが毎秒に敵に与えるダメージ

替えは通常通り遷移条件によって発生し，DNN の切り替えは発生しない．エージェントのステート情報合計 3 個を記録し，合計入力は 17 個になる．二つ目はステートマシンを使用しない，ステート情報は与えない，1 つの DNN で学習させたエージェントである．本手法と比較するニ種類の手法は，すべて同じ行動の制限「攻撃/防御行動は敵が攻撃範囲内にいる場合しか出せない」をかける．全てのエージェントの報酬関数は同一に設定されている．報酬設置は表 1 で表す．

学習アルゴリズムは PPO を選び，全てのモデルを 690 mステップ学習させ，学習過程のデータを比較する．観測パラメータは，学習中キャラクターが毎秒に敵に与えるダメージ (DPS)，キャラクターが毎秒に敵から受けるダメージ，キャラクターが敵を倒す間隔時間の 3 つである．その結果は図 4，5，6 で表す．赤色の線は本手法を用いたモデル，青色の線はステート情報を入力に入れ込んだモデル，緑色の線はステートマシンを使わないモデルである．図 4 は全学習中エージェントが平均毎秒敵に与えるダメージを示す．図 5 は全学習中エージェントが平均毎秒敵から受けるダメージを示す．図 4,5 の横軸がゲーム内の 1 秒を表す．図 6 はエージェントが一体の敵を倒した時間から次の敵を倒すまでの時間を示す．横軸が敵を倒す数，薄い目の線は元のデータで，濃い目の線は移動平均で平滑化したデータである．報酬ではなく観測パラメータを選んで比較する理由は，今回の実験では 3 つの DNN を使うモデルがあり，各モデルの報酬を合わせて時系列に示すのは難しいからである．

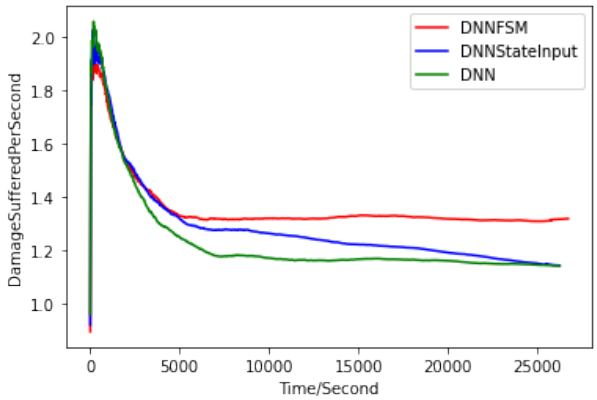


図 5　キャラクターが毎秒に敵から受けるダメージ

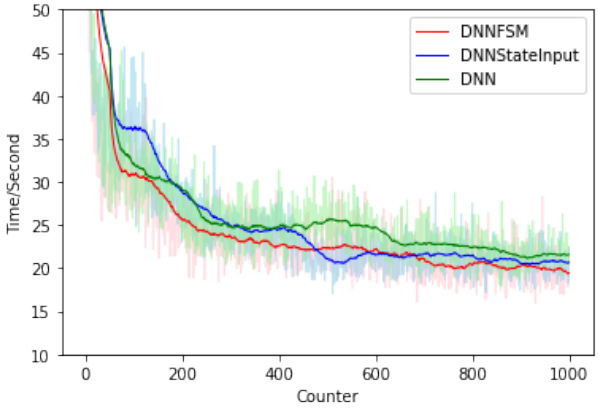


図 6　キャラクターが敵を倒す時間間隔

5.3　結論と考察

上記の結果により，本手法によって作成したキャラクターは与えるダメージと敵を倒す速さの双方の面で優れている，また，実際に学習後のキャラクターの動きを観察する場合，本手法で作成したキャラクター大局的にはステートマシンに沿った動きをしているが，各ステートの実行中には，ステートで要求された動きをしている．例えば，パトロールステートで旋回し，周りを観察して敵を探す．戦闘時，他のモデルが移動行動で回避するに対して，本手法で作成したキャラクターは盾を構える行動で防御し，キャラクターの個性を発揮している．

以上から，今回の実験において，DNN 付きステートマシンを用いたキャラクターは，個性を持ち，かつ単一の DNN を用いたキャラクターより効率よく敵にダメージを与えることを示すことができた．

6.　おわりに

本研究では，深層強化学習とステートマシンの組み合わせについて，DNN 付きステートマシンを提案した．ステートを分割することにより，各ステートのタスクが明確になり，強化学習を人によってコントロールすることが可能になった．今後の課題として，どのようにステートを分割す

ればより良い性能が出ることを検証したい.

参考文献

[1] Badia, Adrià Puigdomènech, et al. "Agent57: Outperforming the atari human benchmark." International Conference on Machine Learning. PMLR, 2020.(2020).

[2] 三宅陽一郎. "ディジタルゲームにおける人工知能技術の応用の現在 (特集 エンターテイメントにおける AI)." 人工知能 30.1 (2015): 45-64.

[3] Damian Isla.Handling Complexity in the Halo 2 AI.GDC 2005(2005).

[4] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602. (2013).

[5] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.(2017).

[6] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018, July). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning (pp. 1861-1870). PMLR.(2018).

[7] Bacon, Pierre-Luc, Jean Harb, and Doina Precup. "The option-critic architecture." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 31. No. 1. 2017.(2017).

[8] Klose, Patrick, and Rudolf Mester. "Simulated autonomous driving in a realistic driving environment using deep reinforcement learning and a deterministic finite state machine." Proceedings of the 2nd International Conference on Applications of Intelligent Systems. 2019.(2019).

[9] Li, L., Wang, L., Li, Y., & Sheng, J. (2021). Mixed Deep Reinforcement Learning-behavior Tree for Intelligent Agents Design. ICAART.(2021).

[10] 上段達弘.「強い」を作るだけが能じゃない！ディープラーニングで 3D アクションゲームの敵 AI を作ってみた.CEDEC2019 (2019).

[11] Juliani, Arthur, et al. "Unity: A general platform for intelligent agents." arXiv preprint arXiv:1809.02627 (2018).

付　　録

A.1 学習の詳細

A.1.1 ハイパーパラメータ

本研究では，Unity 2021.3.8f1 バージョンにて，ML-Agents Release 17 を使用して学習を行った．本稿の比較実験に使われた DNN の学習ハイパーパラメータは表 A·1,A·2 で示す.

A.1.2 ニューラルネットワークの規模

本稿の比較実験に使われた DNN の規模は，表 A·3 で示す.

表 A·1 パトロール，追撃ステート，ステートインプット，ステートマシン不使用 DNN の学習ハイパーパラメータ

ハイパーパラメータ	値
バッチサイズ	128
バッファーサイズ	2048
学習率	0.0003
beta	0.005
epsilon	0.2
lambd	0.95
num_epoch	3
reward_signals:gamma	0.95
reward_signals:strength	1

表 A·2 攻撃ステート DNN の学習ハイパーパラメータ

ハイパーパラメータ	値
バッチサイズ	128
バッファーサイズ	2048
学習率	0.0003
beta	0.005
epsilon	0.2
lambd	0.95
num_epoch	3
reward_signals:gamma	0.9
reward_signals:strength	1

表 A·3 各 DNN の重みの数

DNN	重みの数
パトロール，攻撃，追撃ステート	72973
ステートインプット	73741
ステートマシン不使用	72973