DQN を用いたシューティングゲーム AI の制作

○中川翔太　永田裕一（徳島大学）

Create a shooting game AI using DQN

* S. Nakagawa and Y. Nagata (University of Tokushima)

Abstract－ Game AI uses deep reinforcement learning to allow a computer to automatically learn the rules while playing a game without being programmed with knowledge of the game. Therefore, it is expected to lead to research on AI that is more versatile than AI for specific games. For example, deep reinforcement learning has been applied to various video games, and attempts have been made to create computer players that can clear these games. In this study, I use DQN to create an AI for a shooting game. The agent learns from easy to gradually difficult environments. In particular, the agent learns to focus on the behavior of avoiding enemy attacks.

Key Words: Artificial Intelligence, Neural Networks, Learning

1. はじめに

ゲーム AI では囲碁，将棋など特定のゲームを研究対象とし「強くて負けない」アルゴリズムが開発されてきた．しかしその場合，研究結果がゲームのルールに依存し応用が限られている．そこで，より汎用的なゲーム AI の研究として機械学習が注目されている．

近年では，深層強化学習の手法の 1 つである DQN は，家庭用ゲーム機 Atari2600 上の 49 種類のゲームに応用され，そのうち 29 のゲームでプロのゲーマと同等以上の性能を有するプレイヤの制作に成功している．DQN には Rainbow などの様々な改良版があり，これらゲームに関しては，さらに強力なプレイヤの制作が可能となっている．他にも，「深層強化学習による東方 AI」[1]（以下，東方 AI）のように深層強化学習を市販のゲーム「東方紺珠伝」に応用してゲームをクリアする AI を制作することを目的とした実験も行われている．しかし、東方 AI の実験の結果で得られたゲーム AI の性能は人間のプレイヤに劣っているとされている．

そこで，東方 AI を基に，自作のシューディングゲームの環境に DQN を用いてシューティングゲーム AI を制作する実験を行ったところ，ゲーム AI は敵の攻撃を避けることなく，シューティングゲームを攻略することができない結果が得られた．このことから多数の玉が飛び交うゲームにおいては，DQN では玉をかわすこと自体が難しいタスクであると考えられる．

本研究では自作のシューティングゲームの環境と深層強化学習手法の一つである DQN を用いてシューティングゲーム AI の制作を行う．エージェントはシューティングゲームにおける敵の弾を避ける行動に注目した学習を行う．

2. 研究目的

本研究では，シューティングゲーム AI を制作することが目的である．しかし，自作のシューティングゲームの環境に DQN を用いて学習させたエージェントは敵の弾を避ける行動を学習しなかった．そこで，多数の弾を避ける行動を学習することが難しいことがわかったので，弾を避ける行動を学習させることに焦点を当てた実験を行う．この実験を通してエージェントは弾を避ける行動を学習することが期待される．

3. 関連研究

3.1. 強化学習

強化学習は，機械学習の研究分野の一分野であり，データから自動的に規則を獲得することができ，音声認識や自動識別など，多くの分野で利用されている．強化学習は，あらかじめ良い状態と悪い状態を決めておき，経験を基に試行錯誤しながら最適な方策を獲得するための枠組みである．

強化学習の代表例として，Q 学習（Q-Learning）がある．Q 学習では，現在の状態s_tのとき行動a_tを選んだ結果，報酬r_{t+1}と次の状態s_{t+1}を観測したとする．この時，次の行動は行動価値関数が最大の値となる行動aを選ぶ予定として，更新式は以下のように定義される．

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right) \quad (1)$$

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a)) \quad (2)$$

仮に行動価値関数$Q(s, a)$の正しい値が求まるならば，

$$Q(s_t, a_t) = r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \quad (3)$$

という関係式が成り立つ．ただし，学習途中では正しい行動価値関数が求まっていないため等号は成り立たない．このとき，両辺の差である $r_{t+1}+$

$\gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$ を TD 誤差と呼び, Q 学習では TD 誤差が 0 になるように行動価値関数を更新していく．ここで, αは学習率を表し, γは割引率を表す．学習が収束していたとすると，TD 誤差は 0 になるはずである．

3.2. 深層学習

深層学習（Deep Learning）は, ニューラルネット（Neural Network）の学習手法を発展させて機械学習手法の一分野である．深層学習では，従来のニューラルネットでは扱うことのできなかった大規模なデータに対して学習を実現することができるため，様々な事例へ応用されている．例えば，画像認識の分野だと，与えられた画像に何が映っているかを自動で認識することができる．

深層学習には，ニューラルネットワークの層を深くしたディープニューラルネットワーク（Deep Neural Network）を起点として, 様々な派生がある. ここでは，本研究で使用する畳み込みニューラルネットワーク（Convolutional Neural Network）について記述する．

従来のニューラルネットワーク（DNN）と違い, CNN は 3 次元という形状を維持したまま処理を行うため，空間情報を考慮した処理が可能になっている. DNN は全結合層の NN が用いられているため，画像を入力する際に画像の形状を分解して 1 次元のデータにする必要がある．大規模で複雑なデータを処理するためにニューロン数を増やすと，結合数が爆発的に増加し，それに伴い重みや閾値の組み合わせが非常に多くなり，それらを適切に学習することが極めて困難になってしまう．このため，全結合の DNN には適応できるデータや認識制度に一定の限界があった．一方，CNN は画像の特定の特徴を抽出する畳み込み層と，画像をぼかして全体の特徴を抽出するプーリング層を組み合わせて構成されている．畳み込み層やプーリング層ではニューロンは全結合せず，もっと単純な処理を繰り返す形式で構成されているため，大規模で複雑な CNN でも，重みや閾値の学習が可能になる．

3.3. 深層強化学習

深層強化学習は深層学習と強化学習を組み合わせた学習手法である．深層強化学習は，強化学習と学習の対象は同じだが，行動の選択にニューラルネットワークを用いるという点で従来の強化学習と異なる．ニューラルネットワークを用いることにより強化学習では扱えなかった連続値を用いて学習が行える．例えば，テーブル形式の Q 学習では，状態と行動の数だけ行動価値を表す Q 値を配列に格納する形で表現する．この方法で表現する場合状態と行動の組み合わせが多くなるほど Q 値の表現に膨大な記憶容量を必要とする．Q 値の表現が膨大になると，そもそも Q 値の学習が困難になる．この問題を解決する方法として，Q 値の表現とその獲得にニューラルネットワークを用いる方法がある．

深層強化学習の代表例として DQN がある. DQN は, Q 学習における行動価値関数を畳み込みニューラルネットワークに置き換えて近似したものである. DQN では，ある状態sと行動aをニューラルネットワークに入力すると，その状態に対応する行動選択Q(s, a)をニューラルネットワークの出力として得られるようになる．

3.4. 深層強化学習による東方 AI

深層強化学習による東方 AI[1]は「東方紺珠伝～ Legacy of Lunatic Kingdom.」というシューティングゲームに深層強化学習の手法である Double DQN と DRQN を組み合わせたものを用いて，入力に画面情報を用いてゲームをクリアする行動を学習させることを目的として実験が行われている．ここでは，30 時間の学習を行うことで，生き残る行動を学習している．しかし，人間のプレイヤと比べると劣っているため数フレーム先の弾幕を予想して動きを組み立てることができていないと記されている．

4. 予備的検討および実験

1 章でシューティングゲーム AI は敵の攻撃を避けないことが問題として取り上げた．シューティングゲーム AI として弾を回避する行動が求められるため，本研究では弾を避ける行動を学習させることを目的に実験を行う．本研究では，いきなり弾を避ける行動を学習させることは難しいと考えられるため，シューティングゲームより簡単なタスクでエージェントが学習できるかを確認して，そこから徐々にシューティングゲームへと設定を近づけて実験を行い，最終的に弾を避けるシューティングゲーム AI の制作を試みる．また実験の中でいくつかの工夫を施すことで性能の向上を図っている．以下では実験ごとの内容，環境設定，工夫，結果を示す．今回は，3 つの実験について述べた後，まとめて考察を行う．

4.1. 環境設定

自作のシューティングゲーム環境に DQN を用いて学習させた実験で使用した設定について説明する．本研究では，Python の Chainer を使用して実験を行っている．特に記述しない場合，これ以降の実験でも以下の設定で実験を行う．

- 実験で使用する環境は Fig. 1 のように，自機が 1 つといくつかの弾があり，自機と弾が接触することで被弾として扱う．弾の出現位置や速度は目的に合わせて自由に設定できる．
- DQN のニューラルネットワークは Fig. 2 のとおりである．畳み込み層 3 層と全結合層 2 層の 5 層からなり, 活性化関数に ReLU 関数を用いている．

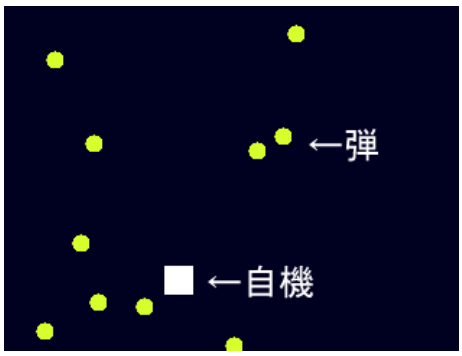

Fig. 1, Game screen of self-made environment

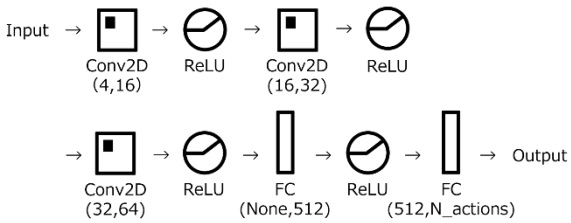

Fig. 2, Neural Network

- 入力はスクリーンショットを用いてゲーム画面を読み取り，それを縦横それぞれ 1/4 に縮小してグレースケール画像に変換した最近の 4 フレームを用いる．
- 出力は上下左右移動の 4 通りと移動なしの 1 通り，合計 5 通りとする．
- 報酬は被弾時に-1 の報酬が与えられる．
- 学習は 1,000 ステップを 1 エピソードとする．エピソード数は実験により異なる．
- 探索方法には ε -greedy 法を用いて，はじめは epsilon =1 からはじまり完全なランダム行動を行い，epsilonの値は直線的に減少していき総ステップの 7 割を超えるところでepsilon = 0.1まで減少する
- Experience Replay を行うための replay buffer は10^4として，5,000 ステップ後から Experience Replay を開始する．ミニバッチのサイズは 200 ステップ，モデル更新間隔は 100 ステップ，ターゲットモデルの更新間隔 2000 ステップとする．

4.2. 実験目的

本研究は，エージェントにシューティングゲームにおける弾を避ける行動を学習させるために，自作の環境では，弾を避ける行動を学習するために最低限必要であると考えられる，自機と弾のみを実装している．最終的にエージェントは与えられた状態から自機と弾を判断し被弾しないような行動を選択することが求められる．今回は実験結果を検証しながら各実験において期待する行動の学習が行われたか確認しつつ随時変更を加えていきより良いモデルに近づけていく．

4.3. 実験 1

まず，弾を避ける行動が学習できなかったため，エージェントは自機を認識できているかを調べるために，確実に学習できると考えられるタスクから実験する．実験 1 はエージェントに動かない弾に当たりに行く行動を学習させることを目的に実験を行う．ここで行動を選択したときに動くのは自機だけであるので，弾に当たりに行く行動を学習できたなら，エージェントは自機を認識できると考えられる．

4.3.1. 環境設定

実験 1 は画面内のランダムな位置に動かない弾が出現し，エージェントは弾に当たることで+1 の報酬を得られるため，弾に当たりに行く行動を学習することが期待される．出現位置や 1 度に表示される数は自由に調整でき，自機が弾に当たると，また別の位置に弾が表示される．学習時間は 2000 エピソードに設定する．

4.3.2. 入力画像の 2 値化

4.1 節の設定でエージェントの学習を行ったところ，エージェントは 1 つの行動だけを選択し続けるようになり，弾に当たりに行く行動をすべての状態において学習できていない．そこで，入力情報を単純にして学習の効率化を図るために，次のような工夫を加える．

入力に使用される各ピクセルの画素値を 0 か 255 の 2 値に変換する．画素値の変換は閾値を決めて，そのピクセルが閾値以上の画素値を持つ場合はそのピクセルに 255 を代入し，閾値より小さい画素値を持つ場合はそのピクセルに 0 を代入する．ここでは閾値を 128 に設定する．

4.3.3. 結果

各ピクセルの画素値を 2 値化することで，エージェントは弾に当たりに行く行動を学習した．Fig. 3 は，工

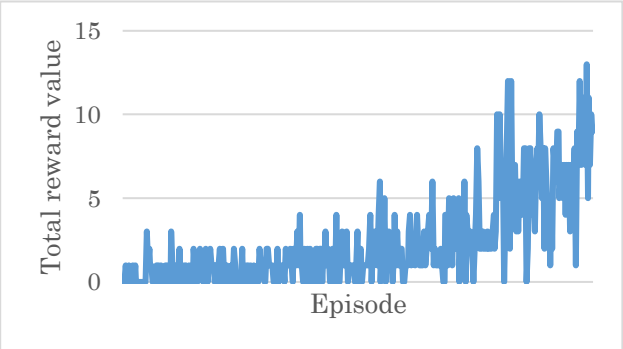

Fig. 3, Experiment 1: Graphs during learning after adding innovations

夫を加えた後の学習中のエージェントが各エピソードで獲得した報酬の合計値を表したものである. Fig. 3 から，エピソードが進むほど 1 エピソードに獲得する報酬の合計値が多くなっていることが読み取れるため，エージェントは積極的に弾に当たりに行く行動を学習していると考えられる.

4.4. 実験 2

実験 2 は，実験 1 で動かない弾に当たりに行く行動が学習できているため，次に弾が動いている場合に弾に当たりに行く行動を学習させることを目的に実験を行う．実験 2 ではエージェントは弾が動いている場合でも自機を認識しているか調べる．弾に当たりに行く行動を学習できたなら，エージェントは弾が動いていても自機を認識できると考えられる.

4.4.1. 環境設定

実験 2 ではエージェントは，弾に当たることで+1 の報酬を得られるため，弾に当たりに行く行動を学習することが期待される．弾はシューティングゲームのように，画面上辺のランダムな位置とタイミングで出現し，画面下辺に向かって移動していく．このとき，弾の動きは直線的に移動し，速さや向きは一定範囲内でランダムに決定している.

実験 1 の結果から，入力画像は各ピクセルの画素値を 0 と 255 に 2 値化したものを使用する．学習時間は 15,000 エピソードに設定する.

4.4.2. 行動の制限

実験 1 で弾に当たりに行く行動を学習したプログラムの設定を用いて学習を行った場合，Fig. 4 のような学習結果が得られた．Fig. 4 は，実験 1 で弾に当たりに行く行動を学習したプログラムの学習中のエージェントが各エピソードで獲得した報酬の合計値を表したものである.

学習の終了したエージェントをランダムな行動をとらないようにして再度ゲームをプレイさせたところ，ほとんど弾に当たることがなく，Fig. 4 を見ても，学習のエピソードが進んでも報酬の合計値に変化が見られないため，エージェントが弾に当たりに行く行動を十分に学習できていない．これは，弾が動いていて，かつエージェントがゲーム画面内を自由に動けることで状態空間が爆発的に増加いているため，学習が難しくなっていると考えられる．そこで，状態空間を減らすために，行動の出力を左右への移動と移動なしの 3 種類に制限する.

4.4.3. 結果

行動を 3 つに制限することで，学習の終了したエージェントは工夫を加える前と比べて格段に弾に当たるようになった．行動を制限したところ Fig. 5 のように学習のエピソードが進むにつれて報酬の合計値は増加している．このことから，エージェントは積極的に弾に当たりに行く行動を学習していると考えられる．Fig. 5 は工夫を加えた後の学習中のエージェントが各エピソードで獲得した報酬の合計値を表したものである.

4.5. 実験 3

実験 3 は，実験 2 で動いている弾に当たりに行く行動が学習できているため，次に本来のシューティングゲームと同様に動いている弾を避ける行動をエージェントに学習させることを目的に実験を行う.

4.5.1. 環境設定

弾やプレイヤの設定は実験 2 と同様に，弾は画面上辺のランダムな位置から画面下辺に向かって移動していき，弾の動きは直線で移動し，速さや向きは一定範囲内でランダムに決定している．エージェントは行動を左右への移動と移動なしの 3 行動から選択する．実験 2 では被弾時に＋1 の報酬を与えることで弾に当たりに行く行動を学習させたため，実験 3 では被弾時に -1 の報酬を与えることで，弾を避ける行動を学習させる．学習時間は 15,000 エピソードに設定する.

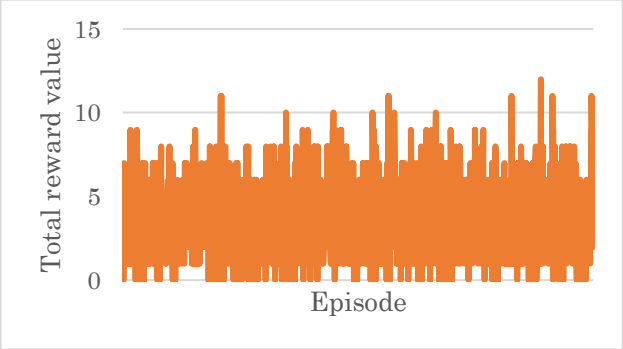

Fig. 4, Experiment 2: Graphs during learning before adding innovations

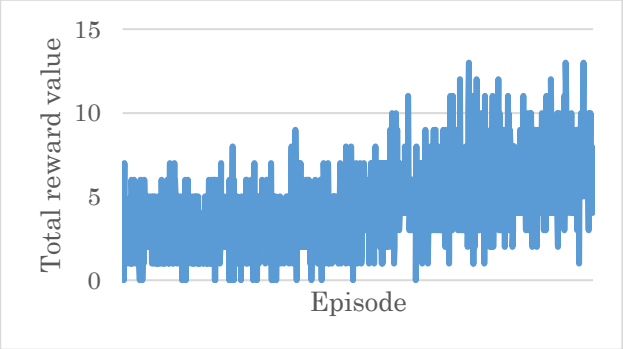

Fig. 5, Experiment 2: Graphs during learning after adding innovations

4.5.2. 8 フレームを 1 枚の画像で表現する

実験 2 で弾に当たりに行く行動を学習したプログラムの設定を用いて学習を行った場合，Fig. 6 のような学習結果が得られた．Fig. 6 は，実験 2 で弾に当たりに行く行動を学習したプログラムの学習中のエージェントが各エピソードで獲得した報酬の合計値を表したものである．学習が進むにつれて報酬の合計値が増えており，弾を避ける動きは確認できた．しかし，1 エピソード中に数回，エージェントは弾に当たる．本来シューティングゲームでは 1 度も被弾したくないため，より性能の良いモデルが求められる．そこで，学習をより効率的に行うために以下の工夫を加える．

これまでは 4 枚の画像を 1 度にニューラルネットワークに入力として渡していたが，実験 3 では観測された最近の 8 フレームを足し合わせて 1 枚の画像にしてニューラルネットワークに入力として渡す．これにより Fig. 7 のような画像が入力に渡され，入力次元を削減しながら最近の 8 フレームの情報を考慮した学習を行うことができると考えられる．1 枚の画像にする方法は以下のとおりである．

ゲーム画面から取得した画像の各ピクセルの画素値を 0 と 1 に 2 値化する．この 0 と 1 に 2 値化された最近の 8 フレームの画像を時間ごとに 2 の指数乗の重みをつけて足し合わせることで 1 枚の画像にする．現在の時刻をt，時刻tで観測された画像をI_tとして，次の式のように各時刻の画像に 2 の指数乗の重みをかけて足し合わせる．

$$I = \sum_{i=t-7}^{t} I_i * 2^{i-t+7} \qquad (4)$$

これにより足し合わせた画像は Fig. 7 のように表現され，1 枚の画像の中で観測された最近の 8 フレームを表現することができる．

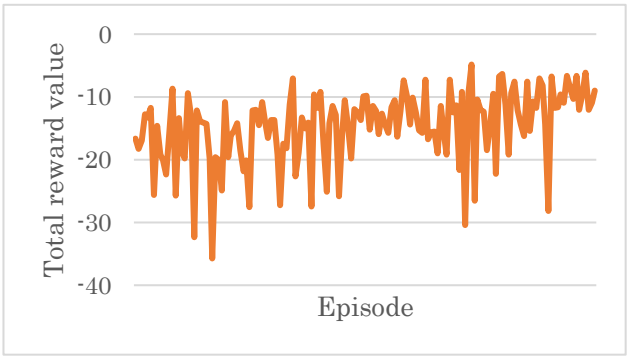

Fig. 6 Experiment 3: Graphs during learning before adding innovations

4.5.3. 結果

学習の終了した従来のモデルと実験 3 のモデルをそれぞれ使用したエージェントに，再度ゲームを 1000 エピソードプレイさせたところ，Table 1 のような結果が得られた．Table 1 はゲームのスコアを表しており，Fig. 8 は Table 1 をグラフにしたものである．Table 1 の「従来」の列は入力情報が 4 フレーム分の成績であり，「実験 3」の列は入力情報が 8 フレーム分を 1 枚の画像にしたモデルの成績である．「報酬」は 1 エピソードあたりの報酬の合計値で，エージェントが弾に 1 回当た

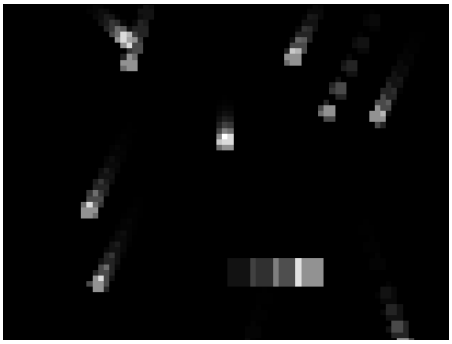

Fig. 7, Representation of the past 8 frames in a single image

Table 1, Game score table

モデル	実験 3	従来
報酬	回数	回数
0	509	294
−1	342	356
−2	115	225
−3	29	78
−4	5	35
−5	0	7
−6	0	5
average	−0.679	−1.245

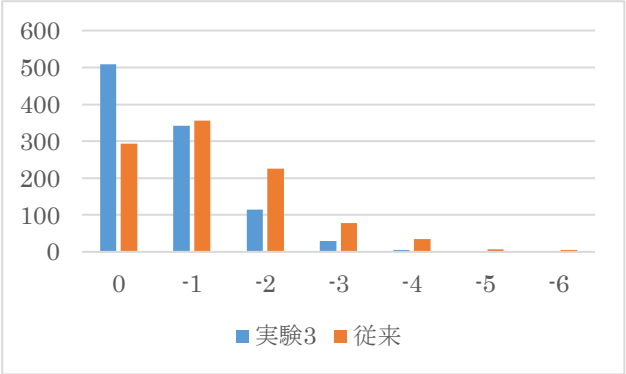

Fig. 8, Graph of Table 1

ると-1 の報酬が加算されていく．「回数」は 1000 エピソード中に「報酬」の値が何エピソード出たかを表している．「average」は 1000 エピソードをプレイして得られた全ての報酬の 1 エピソード当たりの報酬の平均値を表している．

工夫を加える前と後で学習中のグラフに大きな差はみられないが，Table 1 で比較すると，実験 3 のモデルは，弾に当たっていないことを表す報酬 0 の回数が全体のおよそ 50%を占めている．従来のモデルは報酬 0 の回数が全体のおよそ 30%になっている．また弾に当たったことを表す報酬-1 以下では，報酬-1 はほぼ同じ回数で報酬-2 以下は実験 3 のモデルのほうが少なくなっている．さらに average を比べると，従来のモデルは -1.245，実験 3 のモデルは-0.679 となっており，被弾率が半分になっている．このことから，8 フレームを 1 枚の画像で表現して学習したエージェントのほうが評価は高いといえる．

5. 考察

動かない弾に対して当たりに行く実験 1 は，最初は弾に当たりに行く行動を学習しなかったが，各ピクセルの画素値を 0 と 255 に 2 値化する工夫を加えることで，エージェントは弾に当たりに行く行動を学習するようになる．これは画像を 2 値化することで入力される値が単純になり学習するべき対象を認識しやすくなるためと考えられる．

次に動いている弾に対して当たりに行く実験 2 は，実験 1 と同様のプログラムを実装して実験を行うと，エージェントは，ほとんど弾に当たることがなく，エピソードが進んでも報酬の合計値に変化が見られない．そこで，行動を 3 つに制限することにより，エピソードが進むにつれて報酬の合計値が増加し，学習が進んでいると考えられる．これは，選択できる行動が制限されることで遷移する状態が少なくなり，少ない試行で行動を学習できるためだと考えられる．

次に動く弾を避ける実験 3 は，実験 2 と同様のプログラムを実装して実験を行うと，エージェントは弾を避ける行動を選択していることは確認できるが，平均して毎エピソード 1 回は弾に当たる．そこで，入力を最近の 4 フレームの画像から最近の 8 フレームを 1 枚に足し合わせた画像を用いることで，エージェントはより弾を避けるようになった．これは，従来は 4 フレームの画像しか入力の情報として扱わないため行動を選択するのに 4 フレームより前の情報を参照することができないが，実験 3 は 8 フレームの情報を持つ画像を入力に使用しているため，より長い時間ステップの情報を考慮した学習が行えたためだと考えられる．また，画像を 1 枚にすることで，1 枚の画像を学習することと同程度の計算コストですむと考えられるため，学習が効率よく行えたと考えられる．

6. 結論

本研究ではシューティングゲームを模した自作の環境を用意し，まずは確実に学習できると考えられるタスクから学習させ，そこから難しいタスクを学習させることで，弾を避ける行動の学習を行った．実験 1，実験 2，実験 3 で述べた方法を取り入れることでエージェントは敵の弾の回避する行動を学習することができた．しかし，本研究で制作したシューティングゲーム AI は多くの状態で弾を避ける行動を選択することができているが，完全に弾を避けることはできない．シューティングゲームでは 1 度も被弾してはいけないため，今後の課題として更なる性能の向上を目指す必要がある．本研究は， DQN で実験を行ってきたが，さらに性能を向上させるなら，他のゲームで DQN より良い成績を残しているアルゴリズムを使用することが考えられる．また今回は自機と弾でそれぞれ四角と丸の 2 種類の図形しか使用していないが，実際のゲームになると様々な形が使用されているため，性能の向上を図った後に実際のゲームで試す必要がある．

参照文献

1. **能登(@ntddk).** 深層強化学習による東方 AI. 出版地不明 : サークル : 一生あとで読んでろ, 2016.
2. **麻生英樹, ほか.** 深層学習-Deep Learning-. 出版地不明 : 株式会社近代科学社, 2015.
3. **牧野浩二 , 西崎博光.** Python による深層強化学習入門-Chainer と OpenAI Gym ではじめる強化学習-. 出版地不明 : 株式会社オーム社, 2018.
4. **牧野貴樹, 澁谷長史 , 白川真一.** これからの強化学習. 出版地不明 : 森北出版株式会社, 2016.
5. **中出康一.** マルコフ決定過程-理論とアルゴリズム-. 出版地不明 : 株式会社コロナ社, 2019.
6. **小川雄太郎.** つくりながら学ぶ！-深層強化学習-PyTorch による実践プログラミング. 出版地不明 : 株式会社マイナビ出版, 2018.
7. **小高知宏.** 強化学習と深層学習-C 言語によるシミュレーション-. 出版地不明 : 株式会社オーム社, 2017.
8. **−.** 機械学習と深層学習-C 言語によるシミュレーション-. 出版地不明 : 株式会社オーム社, 2016.
9. **久保隆宏.** 機械学習スタートアップシリーズ-Python で学ぶ強化学習-入門から実践まで. 出版地不明 : 株式会社講談社, 2019.
10. **伊本貴士.** ビジネスの構築から最新技術までを網羅-AI の教科書. 出版地不明 : 日経 BP, 2019.
11. **伊庭斉志.** ゲーム AI と深層学習-ニューロ進化と人間性-. 出版地不明 : 株式会社オーム社, 2018.