卒 業 研 究 報 告 題 目

DQNを用いたブロックスのゲームAI

GameAI of blokus using Q-learning

指 導 教 員　　出口 利憲 教授

岐 阜 工 業 高 等 専 門 学 校　電気情報工学科

2020E35 水野　良亮

令 和 7 年（ 2 0 2 5 年 ）　　2 月 1 7 日 提 出

Abstract

In recent years, there have been cases where DQN has been used to perform complex tasks. In this study, we will create a Blocks game AI using DQN.

The Blokus game used in this experiment is a territory capture game for two players. A unique feature of Blokus is that players can only place their own blocks so that they touch the corners of other blocks. DQN is a method that combines Q-learning and neural networks. By using DQN, it is possible to perform reinforcement learning for games with a relatively large number of states, such as Blokus, compared to Q-learning.

The AI has two opponents, one that chooses moves at random and one that plays moves based on basic strategies, with the goal of achieving a win rate of more than 80% against the former and a win rate of more than 50% against the latter.

In the experiment, we tried to increase the winning rate by adjusting the reward settings and batch size for DQN. It was observed that the use of DQN increased the winning rate in matches even with a relatively small number of trials.

The results show a win rate of around 60% against the former and a win rate of over 20% against the latter. Since the target win rate was not achieved, it is desirable to make improvements that will increase the win rate in the future.

目 次

第1章　序論

近年では、囲碁では「AlphaGO」、将棋では「Ponanza」が人間のトッププロに勝利するなど強化学習を用いたゲーム AI が有名になっている。本研究では、ボードゲームとしてブロックスを取り扱い、これのゲーム AI 作成を Deep Q Network(DQN) を用いて行なっていく。

DQN は強化学習の手法の一つである。強化学習は、試行錯誤を繰り返しながら、最適な行動を学習する機械学習の一種である。DQN は、Q 学習とディープラーニングを組み合わせたものである。これにより、テレビゲームのような複雑なタスクを行うことに成功している。本実験で扱うブロックスは比較的複雑なゲームであり、DQN を用いた学習で成果が得られることを期待する。

先行研究として、電気通信大学の大西氏の研究がある。[1)] この研究では、4 人対戦のブロックスのゲーム AI を AlphaZero を修正することで実装している。AlphaXZero はモンテカルロ木探索を基本としたアルゴリズムである。本研究では、2 人対戦のブロックスのゲーム AI を DQN を用いて作成した。

また、今回の研究では、対戦相手としてランダムに手を選択するもの、基本的な方策に基づき手を打つ相手を用意し、前者は勝率 8 割程度、後者は勝率 5 割程度を目指す。実験では、これらの対戦相手に対して、学習のバッチサイズを変更することや、報酬設定を変更することによる勝率の上昇を図る。

第2章　実験で使用した技法

2.1　機械学習 [2)]

機械学習（Machine Learning）とは、コンピュータに大量のデータを読み込ませ、データ内に潜むパターンを学習させることで、未知のデータを判断するためのルールを獲得することを可能にするデータ解析技術である。機械学習の手法は「教師あり学習」と「教師なし学習」、「強化学習」の3つに分けることができる。

1. 教師あり学習

   教師あり学習とは、読み込んだデータから「入力と出力の関係」を学習させることで、データ間の関係を認知させる学習手法である。学習データには事前に「正解」ラベルが付与されている。与えられた学習データの中で、「正解」ラベルが付与されているデータの特徴と、それ以外のデータの特徴をコンピュータが自動で識別し、識別力を向上させていくことで、入力された値に対して「正解」になるデータを出力することができるようになる。

2. 教師なし学習

   学習データにラベルを付与せず、データセットのパターンからデータの関係を認知させる学習手法である。2012年にGoogle社が、教師なし学習により猫を認識できるAIを開発したことが大きなニュースになった。Web上の画像や動画を1週間読み取り続けるうちに、AIが自律的に「猫」というものを認識することが可能になった。このように、与えられたデータからコンピュータ自身がパターンを見出し、未知のデータに対しても予測や識別を行うことを可能にするのが、教師なし学習である。

3. 強化学習

   強化学習は、正解を与える代わりに将来の価値を最大化することを学習するモデルである。囲碁のように、人間が必ずしも正解を予測することが可能なわけではない場合でも学習できるので、人間を超える力を身につけることが期待されている。

2.2　ニューラルネットワーク [4)]

人工ニューラルネットワークは、生物の神経細胞および神経細胞のネットワークの機能をシミュレートすることによって、さまざまな入出力関係を実現する計算機構のこと

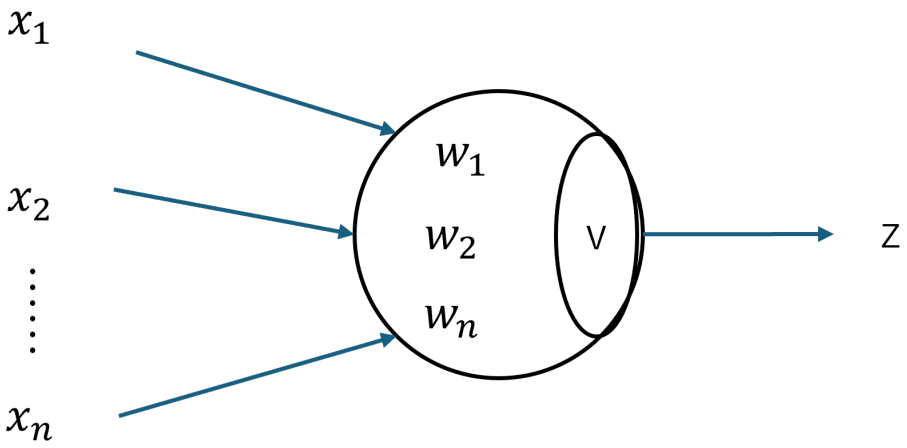

Figure 2.1　Neuron.

である。人工のニューラルネットワークは、生物の神経回路網をシミュレートすることで計算を実現する。Figure 2.1 に人工ニューロンの模式図を示す。ここで、$x_1, \ldots, x_n$ は入力、$w_1, \ldots, w_n$ は重み (結合荷重)、v はしきい値、z は出力 ($z = f(u)$)、f は伝達関数 (活性化関数) である。

人工ニューロンは別の人工ニューロンから情報を受け取り、適当な処理を施したうえで、次段の人工ニューロンへと情報を伝達する。これを繰り返すことで、全体としてある処理を行う、人工のニューラルネットを構成する。

ニューラルネットは、入力となる情報を与えられると、ネットワークを構成する個々の人工ニューロンがそれぞれ計算を行い、後段の人工ニューロンへ計算結果を伝達する。最後に、ネットワークの出力部に配置された人工ニューロンが最終結果を出力する。

Figure 2.2 にニューラルネットの模式図を示す。このとき、ある入力に対して期待した出力が現れるようにするためには、人工ニューロンの内部パラメータを適切に設定する必要がある。このように、内部パラメータを適切に設定することでニューラルネットワークの動作を期待するものに整える操作を、ニューラルネットワークの学習と呼ぶ。

ここで伝達関数 (活性化関数) には様々な関数が用いることができる。次項ではその活性化関数について触れる。

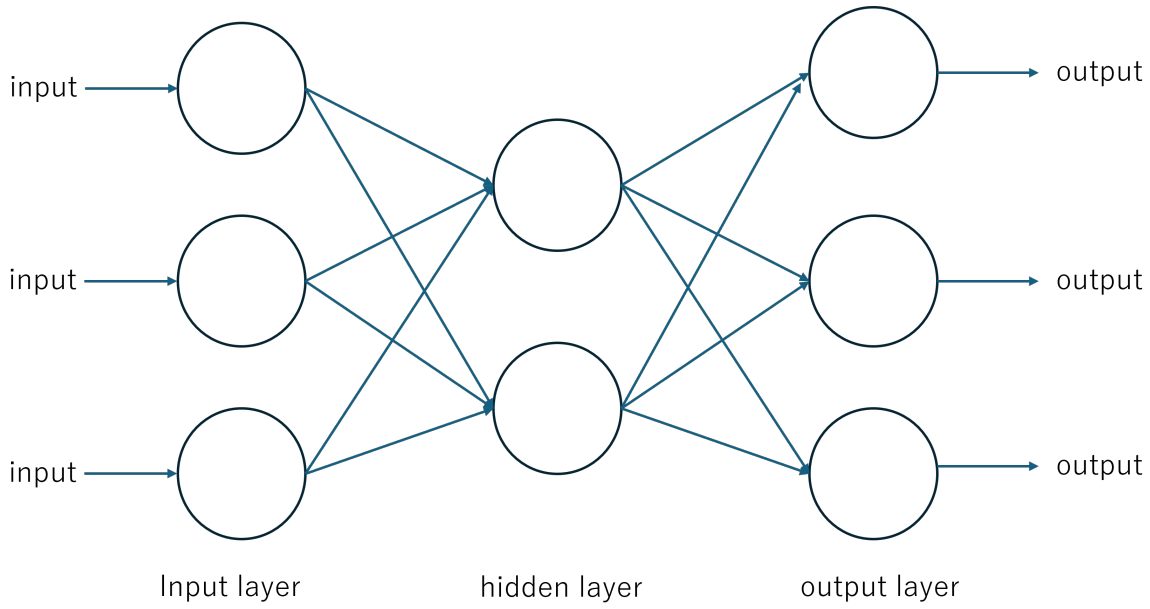

Figure 2.2　Neural Network.

2.2.1　活性化関数 [4)]

人工ニューロンは、生物の神経細胞と同様に、複数の入力と一つの出力を有した計算素子である。各入力には数値が与えられる。

与えられた数値は、それぞれの入力ごとにあらかじめ決められたパラメータである重みまたは結合荷重と呼ばれる定数と掛け合わされ、その結果が合算される。さらに、この合計値から閾値と呼ばれる定数を引き算する。この計算結果を適当な関数によって変換した結果が出力の値となる。

この関数を、活性化関数と呼ぶ。活性化関数として ReLU を紹介する。

ReLU

ReLU 関数は以下の定義式で示される活性化関数である。ReLU のグラフを Figure 2.3 に示す。ReLU を使うと、勾配消失問題が起きにくいという効果がある。

$$\mathrm{ReLU}(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ x & \text{if } x > 0. \end{cases}$$

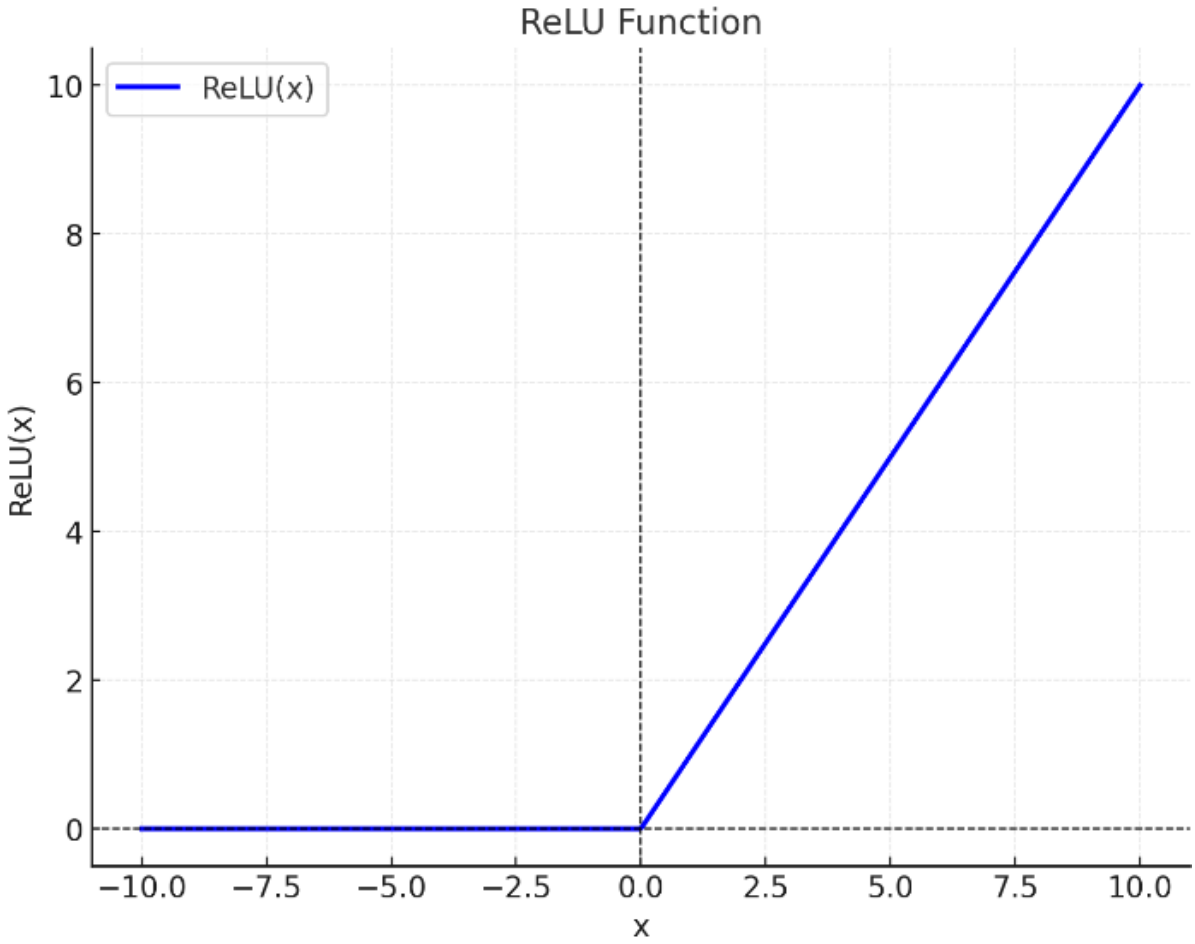

Figure 2.3　ReLU.

2.2.2　バックプロパゲーション [4)]

一般にニューラルネットの学習とは、お手本となる入出力関係を記述した複数の学習データを用いて、ある入力データに対して期待する出力値が出力されるようにニューラルネットのパラメータを調節する操作のことを言う。バックプロパゲーションは、階層型ニューラルネットの学習を効率的に進めることができるアルゴリズムである。

バックプロパゲーションによる階層型ニューラルネットの学習においては、はじめに、入力データをネットワークに与えて、ネットワークの出力値を計算する。このとき、学習が完成していないネットワークでは、出力値には期待する値とは異なるものが表われる。ここで、期待する値とネットワークの出力との差を、誤差であると考える。そこでバックプロパゲーションでは、誤差が改善されるように、ネットワークのパラメータを調整する。Figure 2.4 にバックプロパゲーションの模式図を示す。

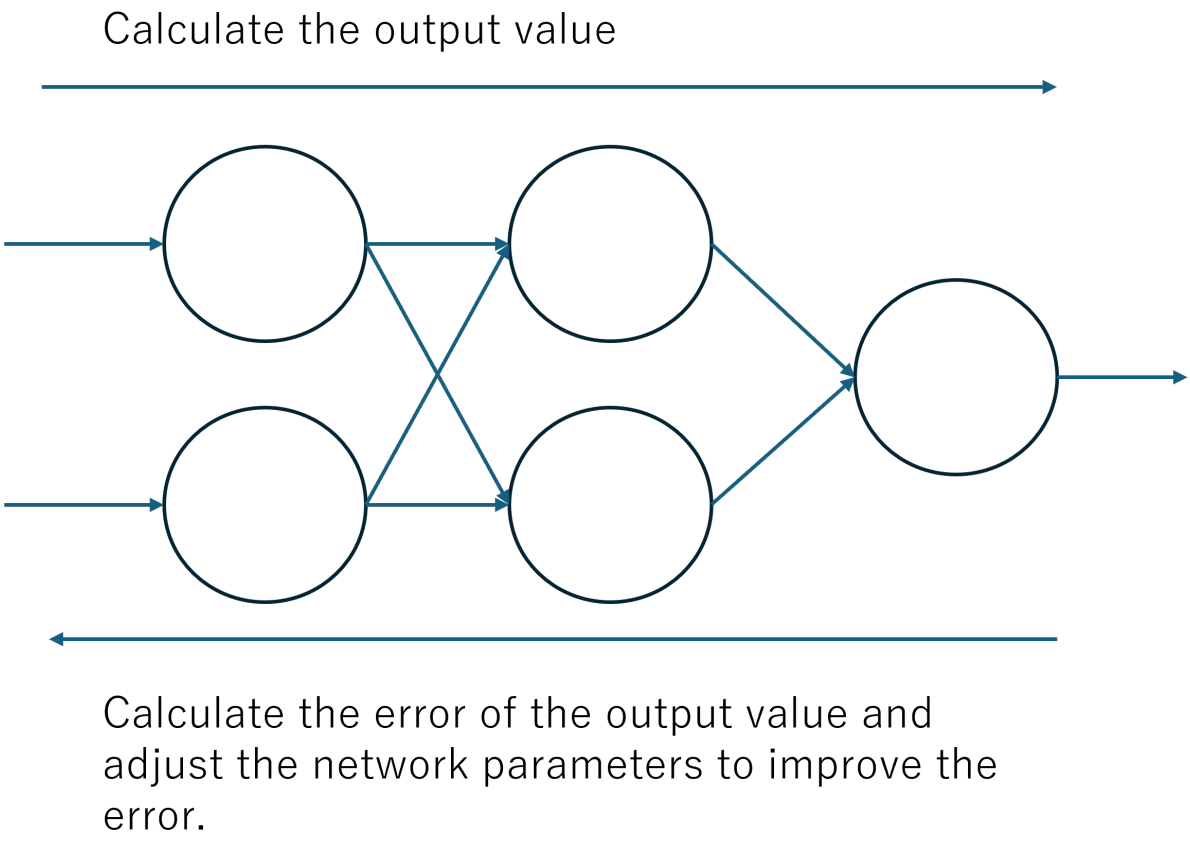

Figure 2.4　Backpropagation.

2.2.3　損失関数 [7)]

機械学習における損失関数（Loss function）とは、「正解値」と、モデルによる出力された「予測値」とのズレの大きさを計算するための関数である。

この損失の値を最小化／最大化することで、機械学習モデルを最適化する。損失関数にはさまざまなものがあるが、特に平均二乗誤差（MSE：Mean Squared Error）が有名である。

平均二乗誤差 (MSE)[8)]

統計学／機械学習における平均二乗誤差（MSE：Mean Squared Error）とは、各データに対して「予測値と正解値の差（＝誤差)」の二乗値を計算し、その総和をデータ数で割った値を出力する関数である。

MSE は、最も一般的な損失関数として使われるだけでなく、主に回帰問題における出力層の評価関数としても用いられる。いずれの関数から出力される値も、0 に近いほどより良い。

2.2.4　深層学習 [4)]

深層学習またはディープラーニングは、ニューラルネットによる機械学習の一種である。深層学習が対象とするニューラルネットは、一般のニューラルネットと比較して大規模で複雑であるという特徴がある。

深層学習の技術を用いると、ニューラルネットを用いて大規模なデータを学習することが可能となる。深層学習は 21 世紀に入ってから発展してきた。主な理由を以下にまとめる。

- ハードウェア技術の発展

  CPU の高速化やマルチコア化、GPU を一般の計算処理に用いる GPGPU 技術の開発、メモリの大容量化など
- 大規模データ (ビッグデータ) の利用可能性

  インターネットの発展、IoT 技術の進展
- ニューラルネットの学習技術の向上

  出力関数の工夫や誤差評価法の改善などによる、学習技術の向上
- ニューラルネットの構造上の工夫

  畳み込みニューラルネットや自己符号化器、あるいは LSTM や GAN といった、ニューラルネットの構造に工夫を施したネットワークの提案

以下では、深層学習でよく利用される畳み込みニューラルネットについて特徴を説明する。

畳み込みニューラルネット (Convolutional Neural Network)[4)]

畳み込みニューラルネット (Convolutional Neural Network; CNN) は、人間の視覚神経系の構造にヒントを得たニューラルネットである。

畳み込みニューラルネットは画像認識の分野でその有用性が示され、その後、他分野への応用が進められている。畳み込みニューラルネットは多層で大規模なネットワークでも学習が可能であるため、深層学習の実現方法として用いられる。

次のページの Figure 2.5 に、二次元データを受け取り、その識別結果などを出力する畳み込みニューラルネットの構造を示す。

Figure 2.5 に示すように、畳み込みニューラルネットは、畳み込み層 (convolution layer) とプーリング層 (pooling layer) が交互に処理を行う構造をもった、多階層の階層型ニューラルネットである。

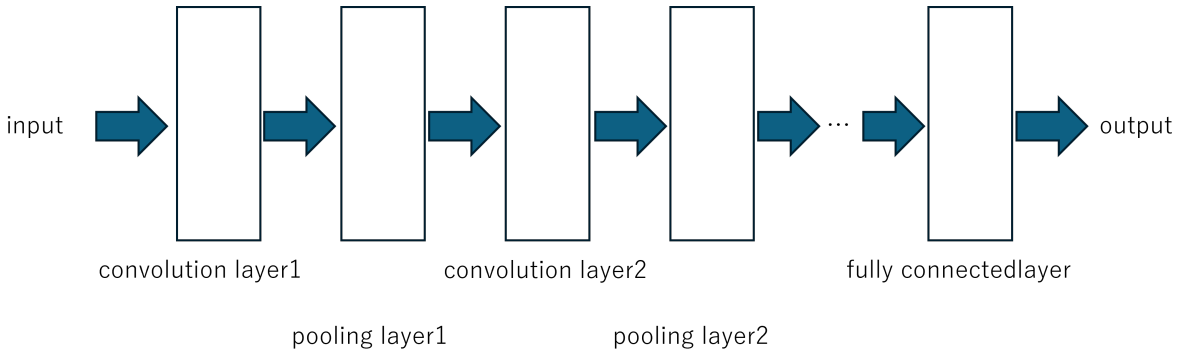

Figure 2.5　Structure of CNN.

畳み込み層では、入力の特徴を抽出する。また、プーリング層では、入力画像の位置ずれや回転などに対応する処理を行う。畳み込みニューラルネットでは畳み込み層とプーリング層を複数積み重ねて、最後に全結合の階層型ニューラルネットを用いて画像認識結果などを出力する。以下では、その働きを説明する。

まず、畳み込み層について説明する。畳み込み層は、画像処理における画像フィルタの挙動をニューラルネットとして実装した情報処理機構である。畳み込み層では、前層から与えられた 2 次元データに対して、畳み込み演算 (convolution) を施す。畳み込み演算の働きを説明する。

畳み込み層に入力として与えられた 2 次元画像のある一点について、その周囲の決められた領域の値を取り出し、あらかじめ決められた係数をかけてその和を求め、領域の画素数で割り算する。この操作は、画像処理における画像フィルタの適用と同じ操作である。このとき、ある点の周囲の領域の大きさは、入力の 2 次元画像に比べてごく狭い領域とする。こうして求められた値を、最初に選んだ元画像のある 1 点に対応するフィルタ出力値とする。次に、同様の操作を入力画像の全域にわたって繰り返す。すると、入力 2 次元画像とほぼ同じ画素数を持った出力画像が得られる。この出力画像は、入力画像の特徴を抽出した画像である。以上の操作を畳み込みと呼ぶ。

畳み込み層の働きは、入力 2 次元画像の特徴を抽出することにある。フィルタの係数を適当に設定すると、たとえば画像の縦方向や横方向の成分を抽出したり、画像のなかで図形の縁にあたる部分を抽出したりすることができる。畳み込みニューラルネットの畳み込み層では、フィルタの係数を学習によって求めることで、特徴抽出に必要なフィルタを自動的に獲得することが可能である。

プーリング層の働きは、入力画像のちょっとした位置ずれや回転などの影響を排除す

るために、画像をぼかすことにある。プーリング層を導入することで、画像認識における汎化能力が高まり、より高い認識能力を獲得することができる。

以上のように畳み込みニューラルネットは画像認識への応用を念頭に置いた構造を有している。しかし、入力データに含まれる特徴を抽出するという意味では、画像以外のデータに対する運用も可能である。たとえば、囲碁プレーヤープログラムの AlphaGo では、囲碁の盤面の解析に畳み込みニューラルネットを利用している。

2.2.5　ニューラルネットワークで用いられる手法

ニューラルネットワークを用いた学習では、学習結果が得られるように様々な手法が存在している。

ドロップアウト [5)]

ドロップアウト (Dropout) とは，ディープラーニングの学習中の各エポックにおいて，各層で異なるニューロンをランダムに除去することにより，簡単な処理で強力な正則化が実施できるテクニックである。ドロップアウトの模式図を Figure 2.6 に示す。

ドロップアウトでは，各学習エポックにおいて，同じ割合だけランダムに「ニューロンの間引き」をしながら学習を行う。これにより，元の「間引き無しネットワーク構造」による推論値の代わりに，間引きによって作られた部分ネットワーク群の推論値の平均近似をデータセットへフィットできるようになる。パラメータが多すぎる Deep Neural Network では起こりがちな「共適合 (co-adaption)」を防ぐことにもつながり，データセットへの過剰適合を防ぐことができる。

ドロップアウトは「毎エポックで，(ベルヌーイ分布などで) ランダムに特定の場所のニューロン間の重みを 0 にして無効化する」だけの処理である。ゆえに，実装もきわめて簡単であり，なおかつ計算負荷も低いテクニックである。それでいて，やっかいな過学習を防ぐことができることから，ディープラーニングにおいてこの層は，広く活用されている。全結合層・畳み込み層直後の，各 ReLU 型活性関数の次にドロップアウト層を配置することが，標準的なネットワーク設計である。

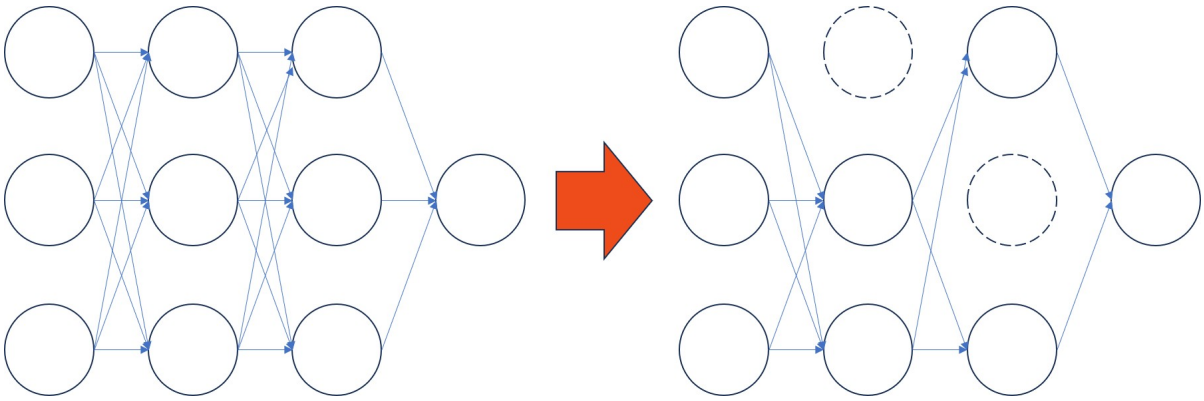

Figure 2.6　Dropout.

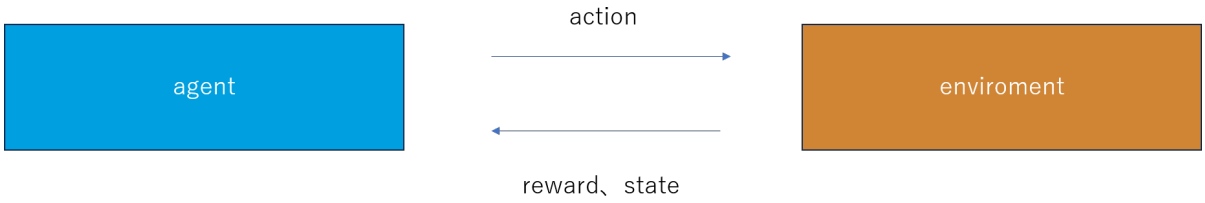

Figure 2.7　Reinforcement learning framework.

バッチ正規化 [6)]

バッチ正規化 (Batch Normalization) は，畳み込みニューラルネットワーク (CNN) の隠れ層において，ミニバッチ内のデータ分布をもとに，各チャンネルごとに特徴を正規化したのち，スケール・シフトを行う，学習の高速化・安定を図る層である。バッチ正規化を各中間層で行うことで，元の DNN の表現力の高さを保ちつつも，学習の収束の高速化と安定化を達成でき，正則化の役割を果たす。

バッチ正規化は提案されて以降，CNN や他 DNN の中間層として，標準的に用いられる層となった。それ以前の CNN では，[畳み込み層 - ReLU(活性化関数) - プーリング層] の 3 種の層で，繰り返しの 1 ブロックを構成する設計が標準的であった。それが，バッチ正規化の登場以降は，[畳み込み層 - バッチ正規化 - ReLU(活性化関数)- プーリング層] の 4 種の層で，1 ブロックを構成する設計が，CNN では標準的となった。

2.3　強化学習 [3)4)]

強化学習では、Figure 2.7 のようにエージェントと環境が相互にやり取りを行う。 エージェントは、何らかの環境に置かれ、環境の「状態」を観測し、それに基づき「行動」を行う。その結果として環境の状態が変化し、エージェントは環境から「報酬」を受け取

ると同時に「新しい状態」を観測する。強化学習の目標は、エージェントが得る報酬の総和を最大にする行動パターンを身につけることである。

2.3.1 Q学習[3)4)]

強化学習を実現する方法としてQ学習がある。Q学習の枠組みにおいては、ある局面における行動選択はQ値に従って決定する。Q値は、学習手続きに従ってあらかじめ決定しておく。実際の行動制御においては、Q値の高い行動を取ることで、最もよい最終結果に至ることができる。Q学習では、次のような手続きを繰り返すことでQ値を学習する。

1. 一連の行動の初期状態に戻る
2. 次の行動をQ値に基づいて選択し、行動する
3. 報酬が与えられたら、報酬に比例した値をQ値に加える。報酬が与えられなかったら、次の状態で選択可能な行動に対応するQ値のうちの最大値に比例した値をQ値に加える
4. 目標状態に達するか、あらかじめ設定した条件に達したら、(1) に戻る
5. (2) に戻る

Q値の更新は以下の式を用いて行われる

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max Q(s_{\text{next}}, a_{\text{next}}) - Q(s,a)\right]$$

ただし、

s: 状態

a: 状態 s で選択した行動

$Q(s,a)$: 状態 s における行動 a に対応する Q 値

α: 学習係数

r: 行動の結果得られた報酬

γ: 割引率

Q 学習の特徴と課題として以下のようなものが挙げられる。

特徴

- 状態遷移と報酬を観測しながら、Q値を少しずつ更新する。これにより、環境の動きに適応しながら学習できる。

- 環境の詳細（状態遷移確率や報酬関数）を知らなくても動作する。エージェントは環境からのフィードバックだけで学習を進めるため、幅広い問題に適用可能である。
- 状態遷移と報酬を観測しながら、Q 値を少しずつ更新する。これにより、環境の動きに適応しながら学習できる。

課題

- 最適な行動を見つけるためには探索が必要だが、探索を多くすると学習が進まない。一方で活用を多くすると、最適でない行動を見逃す可能性がある。このバランスを取るのが難しい。
- 状態と行動の組み合わせごとに Q 値を保持する必要があるため、状態空間が大きくなると計算量とメモリ消費が爆発的に増える。
- 学習が進むためには多くの試行が必要な場合がある。特に大規模な問題や複雑な環境では、収束するまでの時間が長い。
- Q 学習が最終的に学習する行動方針は決定論的（毎回同じ状態で同じ行動を選択する）になりやすい。これが原因で、予測されやすい行動をとる可能性がある。

2.3.2 ϵ-greedy 法[3)4)]

ϵ-greedy 法はシンプルなアルゴリズムである。これは ϵ の確率、例えば ϵ=0.1(10%) の確率で「探索」を行い、それ以外は「活用」を行うという手法である。

「探索」ではランダムに行動を選ぶことで、様々な行動を満遍なく試す。そうすることで、各行動の価値の推定値をより信頼できるようになる。そして、1-ϵ の確率で greedy な行動「活用」を行う。Q 学習では ϵ-greedy 法を用いることで、最適な Q 値を得られるようにする。強化学習では、活用と探索のバランスが重要である。エージェントの経験が増えるにしたがって価値関数の信頼性が上がることを考えると、エピソード数に比例して探索の割合を減らすことは理にかなっている。つまり、初期の段階ではエージェントに多く探索を行わせ、学習が進むにつれて探索を減らす (活用を増やす) ということである。ϵ-greedy 法でこのアイデアを実現するには、エージェントが行動を重ねるにしたがって ϵ を減らすことが考えられる。

Table 2.1　Example of Qtable.

		action			
		a1	a2	a3	a4
state	s1	2	0	0	6
	s2	1	3	0	4
	s3	-4	3	-1	2
	s4	1	4	-3	-7

2.3.3　DQN[3)4)]

DQN は Q 学習とニューラルネットワークを組み合わせた手法である。Q 学習では以下の Table 2.1 のような Q テーブルを作成し、全ての状態と行動に対しての Q 値の保持を行う必要があるため、状態空間が大きくなると Q 学習を行うことは困難になる。それを解決するため、Q テーブルをニューラルネットワークで置き換えるというのが DQN である。

その特徴は、ニューラルネットワークの学習を安定させるために、経験再生 (Experience Replay) とターゲットネットワーク (Target Network) という技術を使っていることである。これらの技術により、DQN はテレビゲームのような複雑なタスクをうまくプレイさせることに初めて成功した。

2.3.4　DQN で用いられる手法[3)]

経験再生 (Experience Replay)[3)]

ニューラルネットワークを使って「教師あり学習」をうまく解いた例は数多く見られる。しかし、ニューラルネットワークを使って「強化学習」の問題をうまく解いた例は、2013 年に DQN が発表されるまでほとんどなかった。教師あり学習のクラス分類の問題などでは、データセットの中身として、画像データと正解ラベルがペアで与えられる。ここで訓練用のデータセットから一部のデータをランダムに取り出す。これをミニバッチ

と呼ぶ。そのミニバッチを使ってニューラルネットワークのパラメータを更新する。ここでミニバッチを作るときには、データに偏りがないように注意する必要がある。Q 学習では、エージェントが環境に対して行動を行うたびにデータが生成される。具体的には、ある時間 t において得られた経験を用いて Q 関数を更新する。ところがこの経験データ間には前後で強い相関がある。つまり、Q 学習では相関の強い (偏りのある) データを使って学習を行っていることになる。

教師あり学習と Q 学習のこの違いを埋めるテクニックが Experience Replay である。Experience Replay ではエージェントが経験したデータを一度「バッファ」に保存する。そして Q 関数を更新する際に、そのバッファから経験データをランダムに取り出して使う。Experience Replay によって、経験データ間の相関が弱まり、偏りの少ないデータが得られる。また、経験データを繰り返し使うことができるため、データ効率がよくなる。

ターゲットネットワーク (Target Network)[3)]

教師あり学習では、学習データに正解ラベルが付与される。このとき入力に対する正解ラベルは不変となる。Q 学習では、$Q(S_t, A_t)$ の値が $R_t + \gamma \max_a Q(S_t + 1, a)$(TD ターゲット) となるように Q 関数を更新する。TD ターゲットは、教師あり学習における正解ラベルに相当する。しかし TD ターゲットの値は、Q 関数が更新されると変動する。ここが、教師あり学習と Q 学習の違いである。この違いを埋めるために、ターゲットネットワークという、TD ターゲットを固定するテクニックが用いられる。ターゲットネットワークの実現方法について説明する。ターゲットネットワークは、強化学習において、特に Q 学習や DQN などのアルゴリズムで使用される技術で、学習の安定性を高めるために使われる。

通常、Q 学習では、Q 値（行動価値関数）を更新する際に、次のステップで得られる最大の Q 値を基に更新を行う。しかし、これをそのまま行うと、ターゲットが学習過程で頻繁に変動し、学習が不安定になることがある。この問題を解決するために、ターゲットネットワークが用いられる。ターゲットネットワークの実現方法について以下に述べる。

1. 2 つのネットワークを作成する:
    - オリジナルネットワーク（学習中のネットワーク）: 現在学習を行っているネットワーク。行動価値関数 $Q(s, a)$ を近似するために用いる。
    - ターゲットネットワーク : オリジナルネットワークのコピー。これが学習の

ターゲットを提供し、オリジナルネットワークを安定させるために使われる。

2. ターゲットネットワークの更新: オリジナルネットワークが学習を進めるとき、ターゲットネットワークは一定間隔でオリジナルネットワークの重みをコピーする。これを「ターゲットネットワークの更新」と呼ぶ。更新の頻度はハイパーパラメータで設定される。例えば、毎回のエピソード終了後に更新したり、特定のステップ数ごとに更新したりする。
3. Q 値の更新: DQN などでは、Q 値の更新において次のような式を使用する。

$$y = r + \gamma \max_{a'} Q'(s', a')$$

ここで、$Q'(s', a')$ はターゲットネットワークから得た Q 値。この式で、次の状態 s' と行動 a' に対する最大 Q 値をターゲットとして利用する。

4. ターゲットネットワークのコピー方法: 通常、ターゲットネットワークの重みはオリジナルネットワークから一定のステップごとにコピーされる。これにより、ターゲットネットワークはオリジナルネットワークの遅延コピーとして機能する。

ターゲットネットワークを用いることで、学習中にターゲットが安定し、学習が急激に変動することを防ぐ。オリジナルネットワークがターゲットネットワークに基づいて学習を行うため、学習の過程でターゲットが急に変化することなく、より滑らかに進行する。ターゲットネットワークは、強化学習の安定性を大きく向上させる重要な技術であり、DQN などの深層強化学習アルゴリズムにおいては欠かせない手法の一つである。

報酬クリッピング (Reward Clipping)[3)]

報酬を -1.0 から 1.0 の範囲に収まるように調整し、報酬のスケールを揃えることで学習を円滑化させること。

2.4 One-Hot エンコーディング[9)]

例えば、「性別」の列に「男性」と「女性」というカテゴリカルデータが存在したとする。このデータは数値ではないので、実際にデータ分析を行うには少し扱いづらい。この時、「男性であるかどうか？」、「女性であるかどうか？」をそれぞれ 1 と 0 で表す。このように数値として扱えないデータを 0 と 1 に数値化する際に用いられる変数がダミー変数である。

One-Hot エンコーディングは、基本的にはダミー変数を用いた変換のことである。例

Table 2.2　One-Hot encoding example.

血液型	A型	B型	O型	AB型
A型	1	0	0	0
B型	0	1	0	0
O型	0	0	1	0
AB型	0	0	0	1

えば、Table 2.2 に示すように血液型のように、数値として扱えない型で表されたデータを数値に変換する際に使用される。

第3章　実験

3.1　目的

この研究では、二人対戦のブロックスでランダムに手を選択する相手と基本的な方策の相手に勝ち越すことができる AI の作成を目指す。

3.2　ゲームの概要

今回の陣取りゲームは 2 人対戦用で、ゲームは Figure 3.1 に示すような画面になる。Figure 3.2 に示す 21 個のピースを 15 × 15 の盤面上に交互に置いていくことで得点を得る。ピースを置くときには回転や裏表を反転させることもできる。最終的にすべてのピースを置くか、相手よりも多くの得点を得たプレイヤーが勝者となる。得点で負けていてもすべてのピースを置くことができたプレイヤーが勝者となるが、両者ともすべてのピースを置き終わった場合には得点で判断する。（当然両者とも置き終わらない場合も得点で判断する。）また、得点も同一の場合には後手のプレイヤーを勝者とする。ピースが一方のプレイヤーがピースを置けなくなった場合には、もう一方のプレイヤーが可能な限りピースを置くまでゲームは継続する。

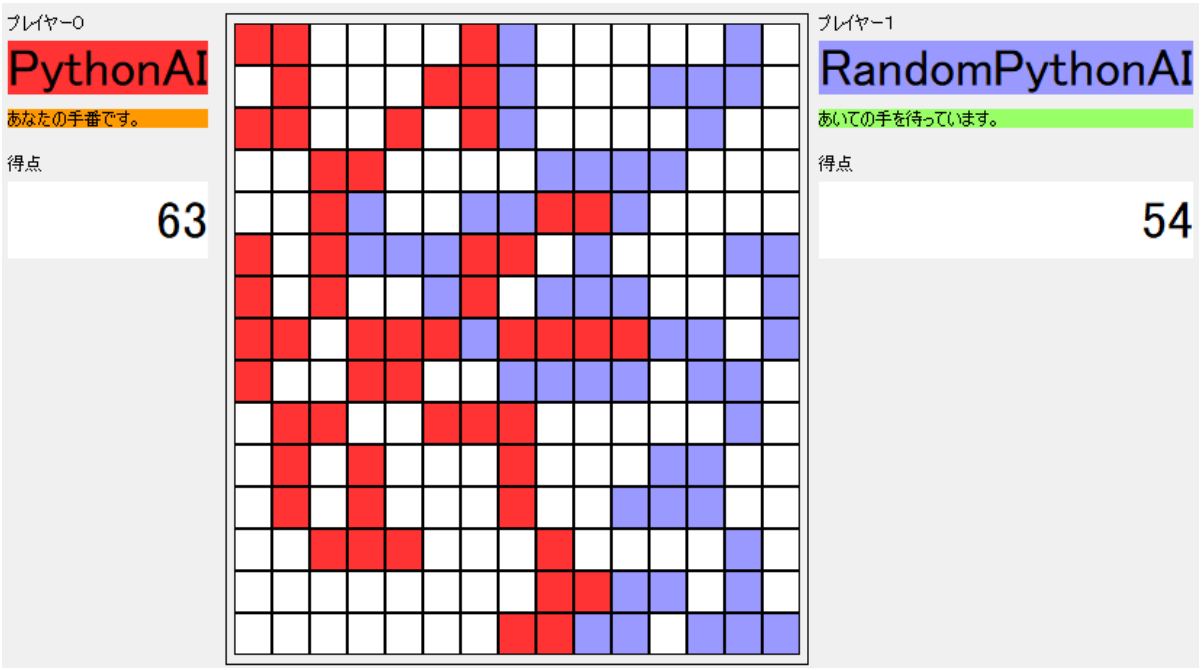

Figure 3.1　Game screen.

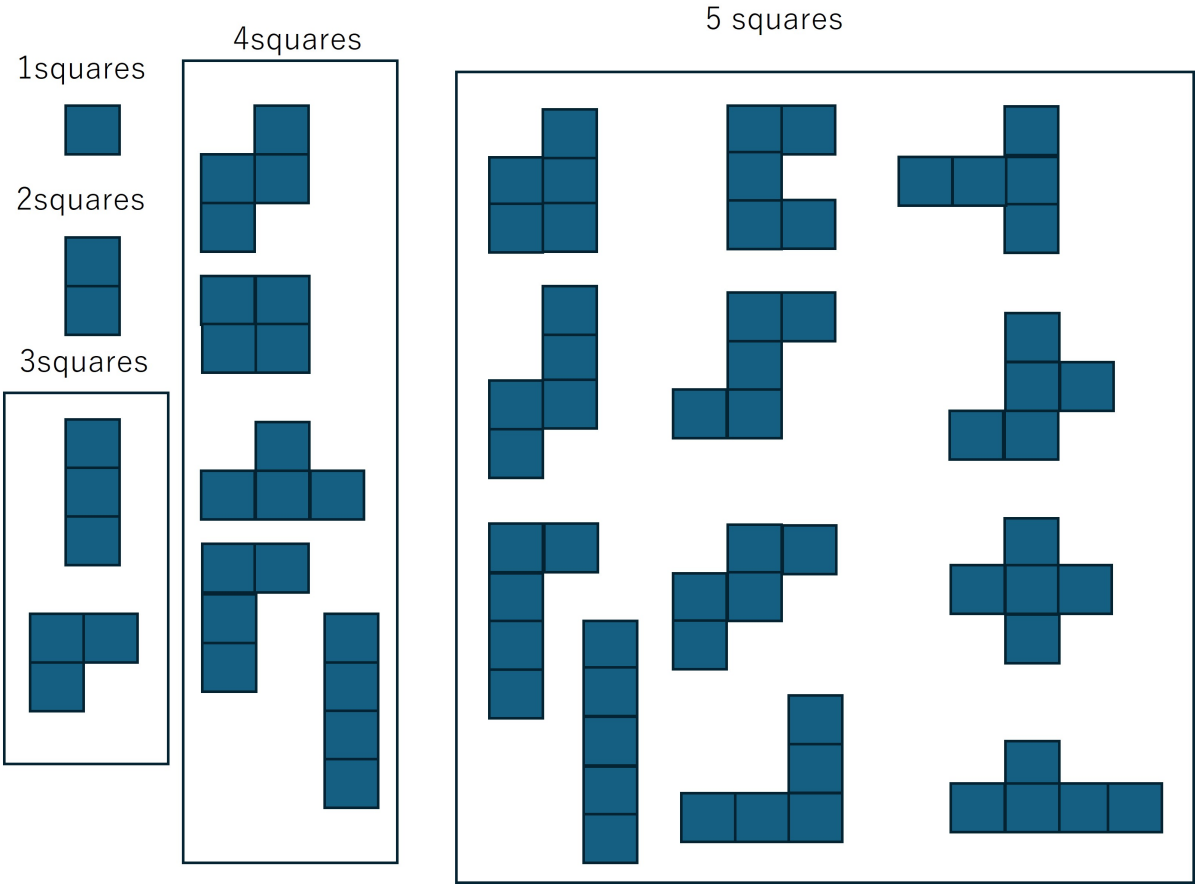

Figure 3.2　Blocks of blokus.

ピースの置き方は Figure 3.3 のような制限がある。

1. 初めのピースはプレイヤーごとに指定された盤面の角にしか置くことができない。
2. 2 つ目以降のピースは自分ピースと角が接するが辺は共有しないように置かなければならない。相手のピースと面や角が接していても問題ない。

得点は置いたピースのマス数といくつの角が接しているかの乗算によって決まる。つまり N マスのピースを M 個の角に接するように置くと M × N 点が加点される。具体例を Figure 3.4 に示す。(1) のように、5 マスのピースを 1 か所の角で接するように置くと 5 点だが、(2) のように 2 つの角に接すると 10 点を得ることができる。

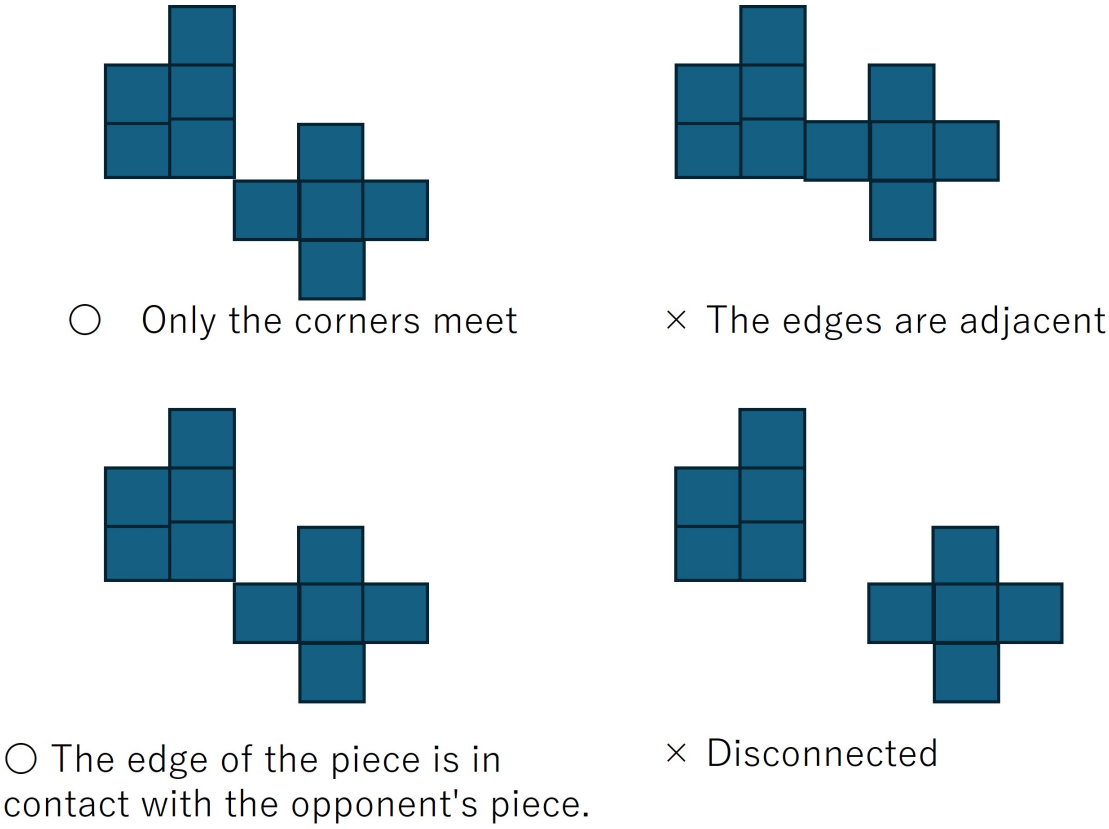

Figure 3.3　Blokus rules.

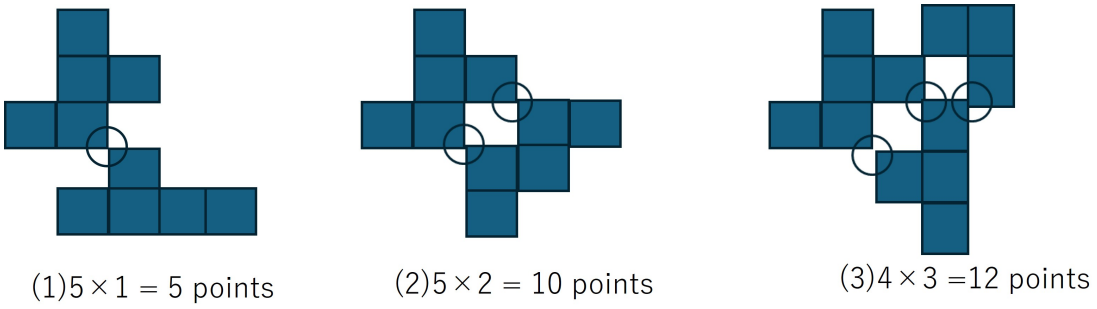

Figure 3.4　How the score is calculated.

3.3　状態

ブロックスの状態は以下のような要素で表されている。

- 盤面の情報

  盤面は 15 × 15 であり、プログラム内では int の二次元配列で表現されている。そして、配列の要素が

  −1: 誰も置いていない

  0: プレイヤー 0 の設置ピース

  1: プレイヤー 1 の設置ピース

となっている。

- プレイヤーが設置可能なピースとその座標
  プレイヤーが盤面に置くことができるブロックとその向きとその座標が HashMap で保存されている。

3.4 行動

プレイヤーは相手と交互に回ってくる手番で、状態で示したプレイヤーが設置可能なピースとその座標の中から何か一つを選択して、行動することになる。設置可能なピースがないときは、pass して手番を相手に回す。

3.5 報酬

今回の実験では、報酬として以下の二つを与えている

- ブロックを置いた時のスコア
- 相手の角にどれだけ接しているか
- 勝敗決定時の報酬

ブロックを置いた時のスコアに関しては、Figure 3.4 のように求めて、それをそのまま報酬としている。次の相手の角にどれだけ接しているかに関しては、一つの角に対して 10 の重みをつけてそれを報酬としている。角が一つなら報酬は 10、角が二つなら報酬は 20 である。勝利したときの報酬としては、100 を与えて、敗北したときには -100 の報酬を与えている。

3.6 対戦相手

対戦相手には、ランダムに手を打つものと、基本的な方策に基づいて行動するもののふたつがある。ここで基本的な方策に基づいて手を打つものについてその概要を説明する。基本的な方策では、スコアを重視して手を選択する。上述したスコアの計算方法に基づいて、今打つことができる手の中から一番得られるスコアの多いものの中からランダムに手を選択する。今回の実験では、この二つの相手で学習を行い、勝率を比較していく。

3.7 DQN を用いた AI の概要

まずは、今回の実験での DQN のハイパーパラメータに関して示す。

- 学習率:1e-4
- 割引率:0.99
- バッチサイズ:64
- 最小学習開始経験数: 1000
- ターゲットネットワーク更新頻度:200 ステップごと
- モデル更新頻度:8 ステップごと
- 初期 ϵ:1.0
- ϵ の減衰率:0.995

学習率は、1e-4 に設定している。学習率は大きすぎると学習がうまくいかず、小さすぎると学習が遅くなる。1e-4 は深層強化学習で一般的に使われる値の範囲内である。

割引率は、0.99 に設定している。0.99 という高い値には、将来の報酬をほぼ現在と同じ価値で評価することを意味する。Blokus は長期的な戦略が重要なゲームだと考えられるので、将来の結果を重視する設定になっている。

初期 ϵ は 1.0 に設定し、その減衰率は 0.995 に設定している。学習の初期段階では、完全にランダムに行動し、減衰率は 0.995 に設定することで緩やかに減少していく。ϵ の値を急激に減少させてしまうと早期に探索をやめてしまい、良い戦略を見つけられない可能性があるため、ϵ の値は緩やかに減少させている。

ターゲットネットワーク更新頻度は 200 ステップごととし、モデル更新頻度は 8 ステップごととする。ターゲットネットワークは頻繁に更新しすぎると、学習が安定しないためモデル更新頻度の 25 倍の値として学習の安定を図っている。

今回は、このハイパーパラメータを基準にして実験を行う。続いて、今回の実験での DQN のニューラルネットワークの構造について解説していく。

今回のネットワークの構造は、15×15 の盤面の状態の情報と、どのピースをどの座標に打つかの行動の情報は初めは違うニューラルネットワークで処理する。まず、盤面の情報は

1. $15 \times 15 \times 1$ の入力を、2 つの畳み込み層で処理
2. 各畳み込み層では、3×3 のカーネルを使用し、32 チャネル→ 64 チャネルと特徴量を増やす

3. バッチ正規化と ReLU 活性化関数を適用する
4. 抽出された特徴は全結合層で 128 次元のベクトルに変換

とする。続いて行動情報は、

- ピースの種類（21 種類)
- 配置座標（x, y）
- 回転情報

これらの情報をピースの種類は One-Hot エンコーディングを用いて 21 次元のベクトルとする。ここに、配置座標、回転情報を示す 0〜7 の数字を組み合わせて 24 次元のベクトルとする。これを入力とし、64 ユニットの中間層で処理する。最終的に、この二つの処理経路からの出力（128 次元と 64 次元）を結合し、全結合層を通して Q 値を出力する。このネットワークの構造の模式図を以下の Figure3.5 に示す。

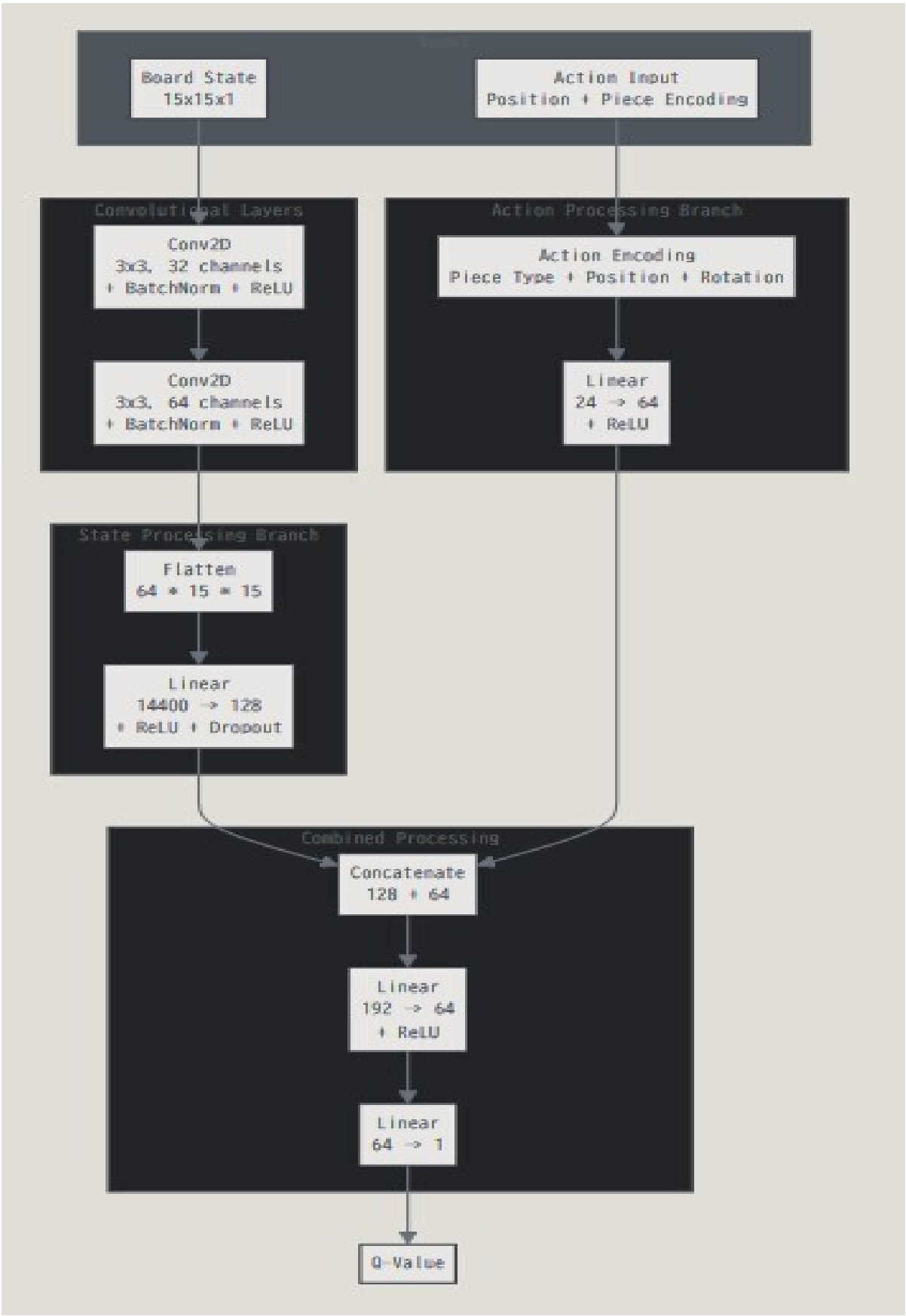

Figure 3.5 The structure of the neural network of DQN.

3.8　対戦結果

3.8.1　ランダムに手を打つ相手との対戦結果

今回の実験では、ランダムに手を打つ相手と基本的な方策に基づいて手を選択する相手の二つと対戦したが、まずはランダムに手を打つ相手との対戦結果の勝率を Figure 3.6 に、どれだけのスコア差がついたかのグラフを Figure 3.7 に示す。

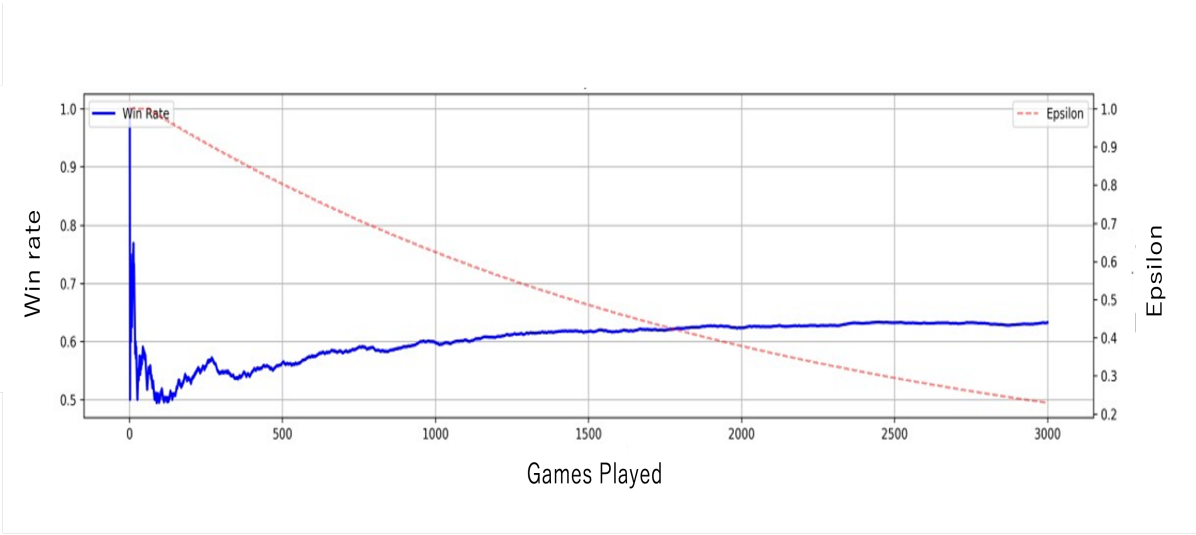

Figure 3.6　Winrate of batch64.

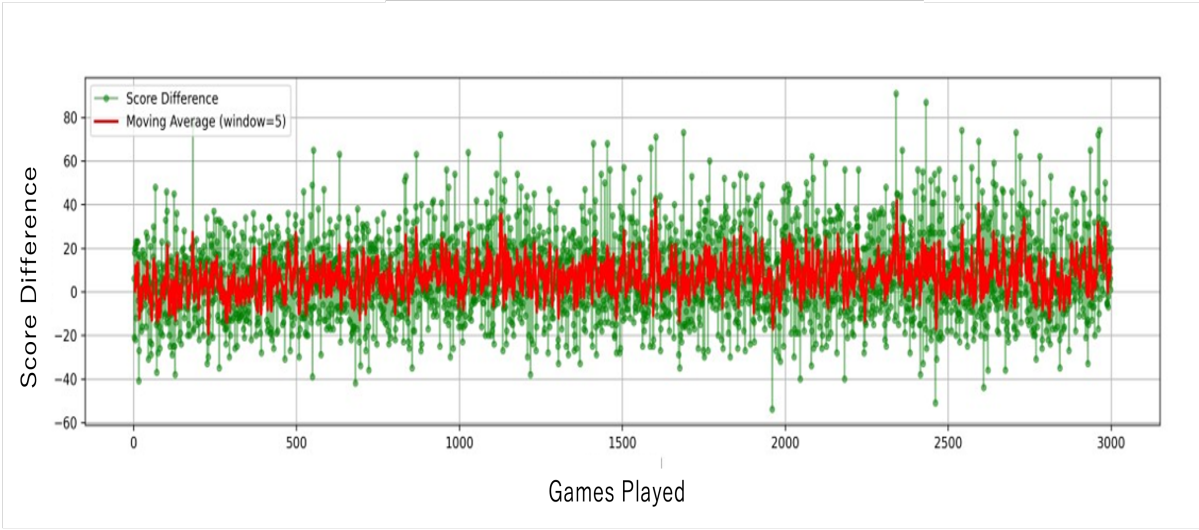

Figure 3.7　Score difference of batch64.

ランダムに手を打つ相手との対戦では、まずは勝率がおおよそ 5 割から推移して最終的には 6 割弱の勝率になっていることがわかる 。最初の方の対戦では、ϵ-greedy 法に基づいて行動するため、ランダムに動作している、そのため勝率は 5 割程度になる。それから徐々にニューラルネットワークを用いての行動へと変化させていき、最終的には 6

割弱に推移していった。

Figure 3.7 を見てみると対戦が後半になるにつれて、大きくスコアの差をつけて勝つことが多くなっているのが、見て取れる。続いて、学習時のバッチサイズを 64 から 512 に増やした時の対戦結果の勝率を Figure 3.8 に、どれだけのスコア差がついたかのグラフを Figure 3.9 に示す。

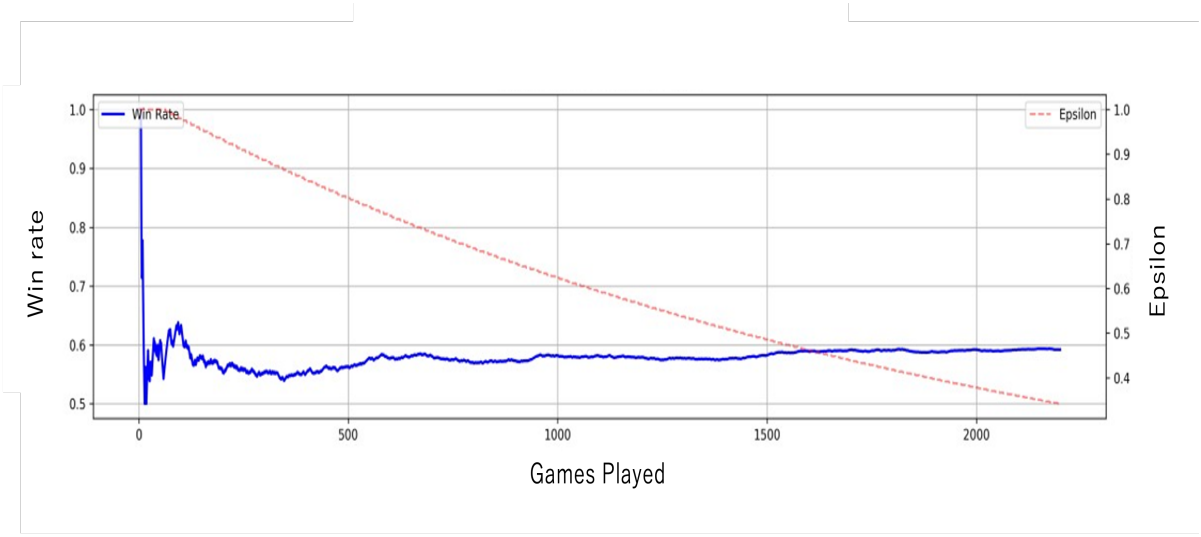

Figure 3.8　Winrate of batch512.

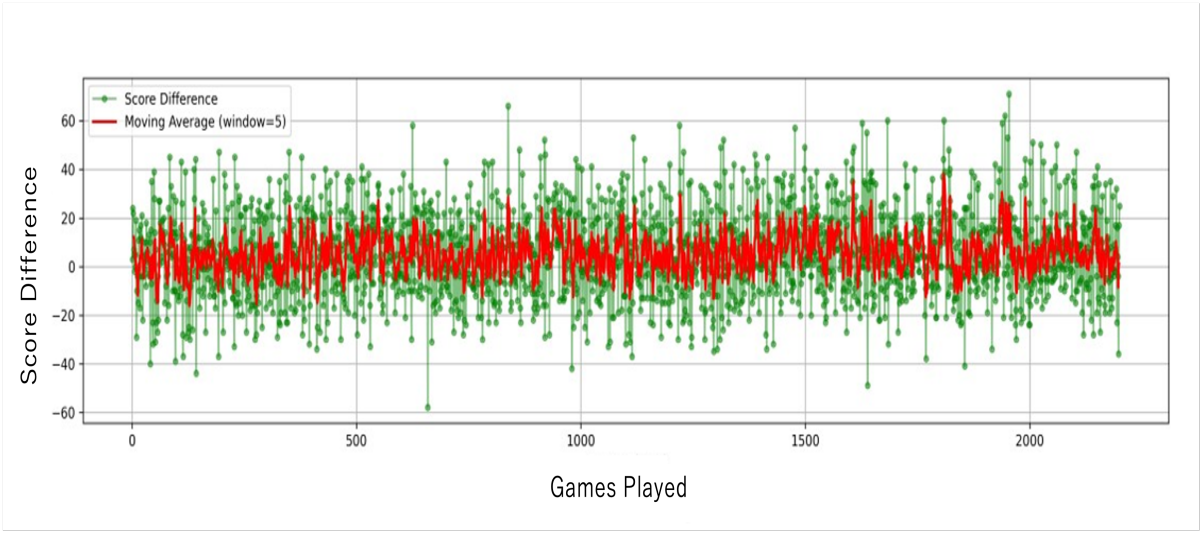

Figure 3.9　Score difference of batch512.

バッチサイズを 512 に増やして学習させてみても、勝率にほとんど変動はなく、おおよそ 6 割という結果になった。

続いて、報酬設定に変化を加えたときの勝率やスコア差を見てみる。ここでは、相手の角を一つ取るごとの報酬の重みを 10 から 5 に減少させ、相手の妨害ではなく、自分のスコアを重視するように報酬設定を変化させた。そのときの勝率が Figure 3.10、スコア差が Figure 3.11 である。

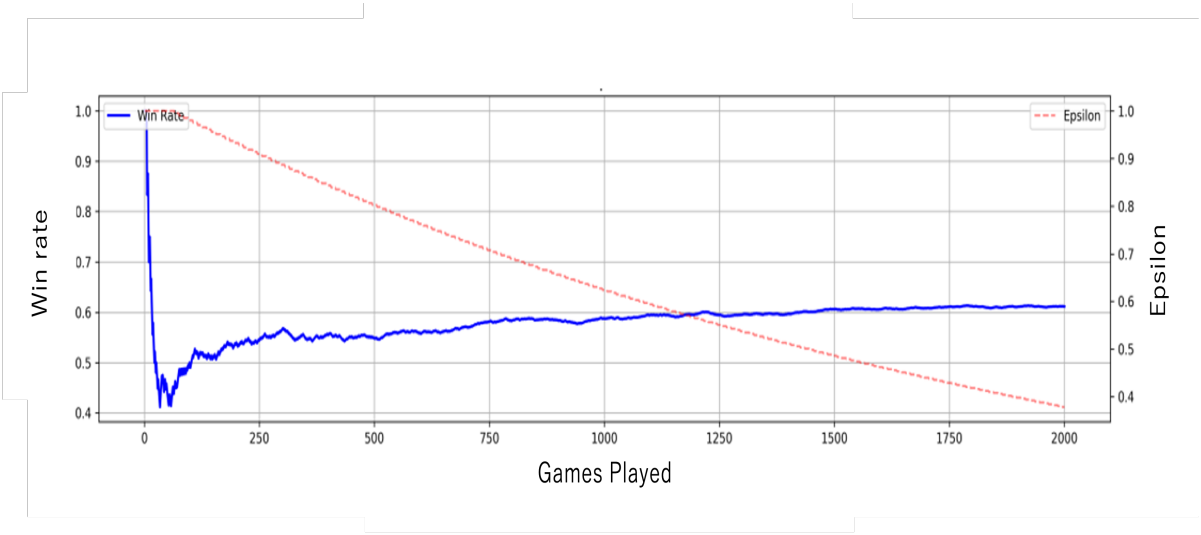

Figure 3.10　Win rate when rewards are changed.

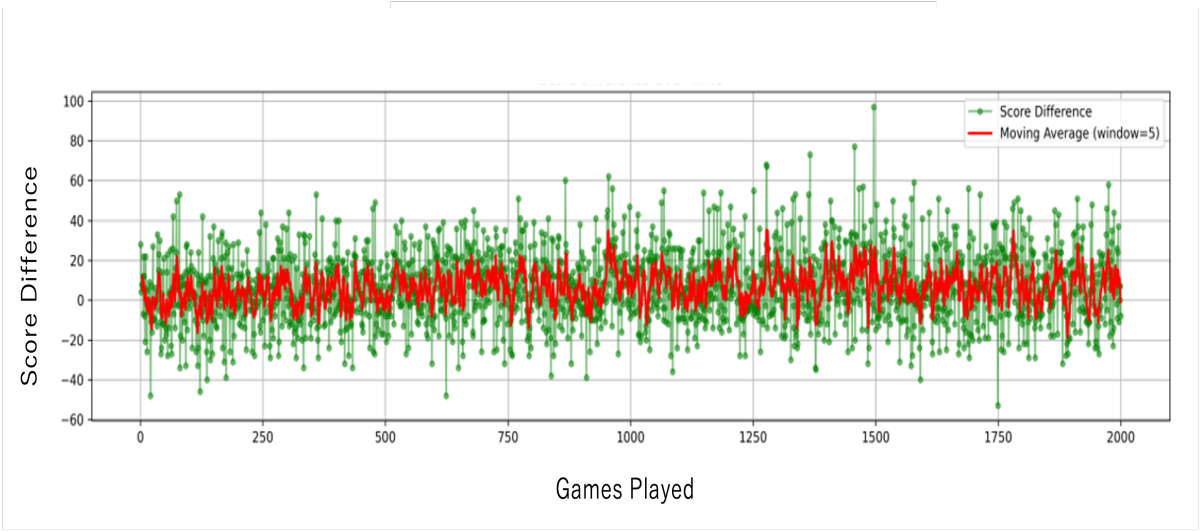

Figure 3.11　Score difference when rewards are changed.

グラフを見ると分かるように勝率は変わらず 6 割程度となった。また、スコア差のグラフを見ると学習を進めていく過程で、スコア差を徐々につけるようになりながら勝利していることが分かる。また、この結果から報酬にこの変化を加えても勝率に大きな変化は見られないことが分かった。

3.8.2　基本的な方策に基づき手を打つ相手との対戦結果

続いて、基本的な方策に基づき手を打つ相手との対戦結果の勝率を Figure 3.12 に、どれだけのスコア差がついたかのグラフを Figure 3.13 に示す。

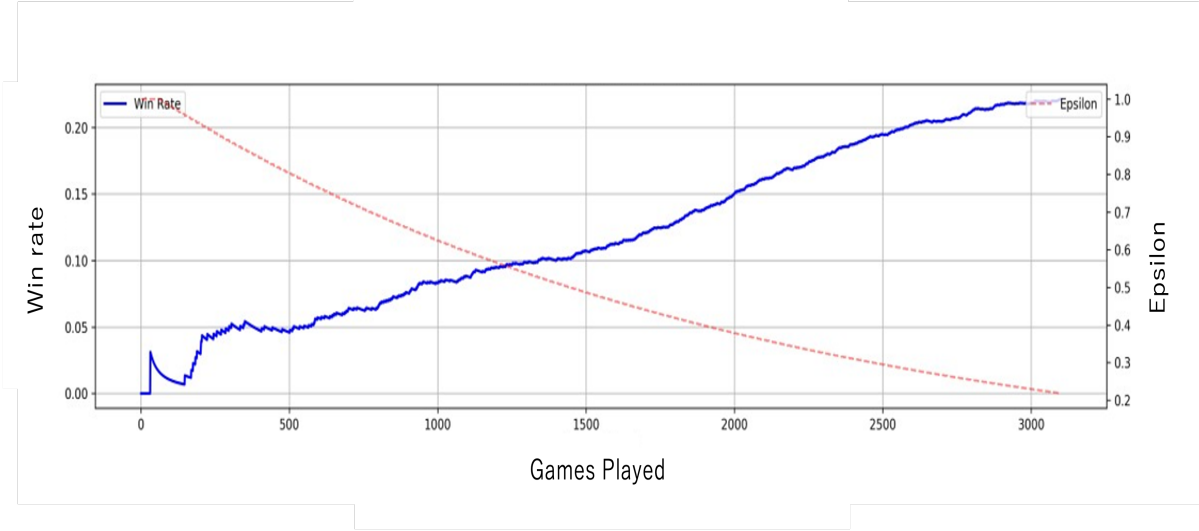

Figure 3.12　Win rate against greedyAI.

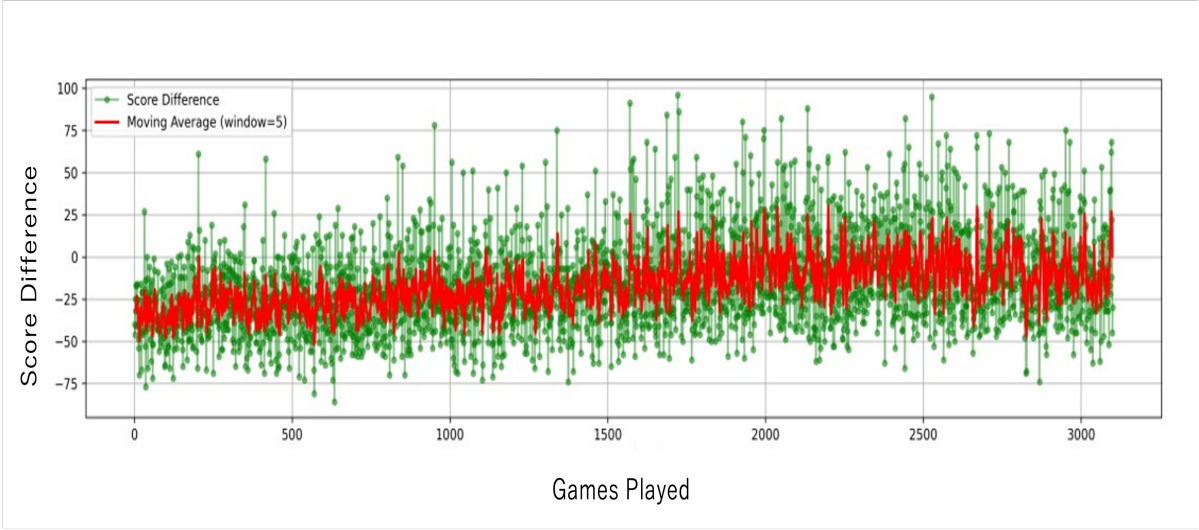

Figure 3.13　Score difference against greedyAI.

基本的な方策に基づき手を打つ相手との対戦では、ランダムに手を選択する相手との対戦とは異なり、自分がランダムに手を選択している ϵ=1.0 の状態では、ほとんど勝つことができないため勝率 0%近くから推移していき、最終的に対戦回数が 3000 回あたりで勝率 2 割弱という結果となった。先ほどのランダムに手を選択する相手と比較して急激に、勝率が上昇している様子が見て取れる。

3.9 考察

今回の実験では、ランダムに手を選択する相手との勝率は 6 割程度、基本的な方策に基づき手を打つ相手との勝率は、2 割程度となった。学習を進めるごとに勝率の上昇は見られるが、当初の目標には及ばなかった。

これらの実験を踏まえて、さらに勝率を上げるためにはどうすればいいのか考察していく。まずは、対戦回数を増やすことが挙げられる。Q 学習及び DQN は大量に試行回数を重ねてその中で得られる報酬から最適な行動を見つけていくアルゴリズムであるためたくさん経験を積ませなければ、まだ行っていない行動の中により高い報酬があることに気づけない。今回は対戦回数を増やす代わりに、バッチサイズを 64 から 512 にするという変更を行うことで勝率に変化が起きないかを見てみたがあまり変化はなかった。やはり、未知の行動を多く取らせるべきなのだと考えられる。

次に考えられる改善案は、報酬の種類を増やしてみることである。今回は、ブロックを置いたときにゲームから与えられるスコアに対しての報酬、自分のブロックを置いたときに相手のブロックの角にどれだけ接しているかに比例した報酬、ゲーム勝敗時の報酬の 3 種類のみであったが、その他にも考えられる報酬として例えば、領土を拡大するために相手のブロックを飛び越えてブロックを配置することが挙げられる。この行動は、相手がブロックを置く領土を奪うことにつながる。

また、同様に状態の種類を増やすことも考えられる。今回は状態として、15×15 の盤面の情報を受けとっているが、それのみではなく相手がまだ使っていないブロックの種類や、自分が使っていないブロックなどの情報も状態として受け取ることでより勝率の上昇につながると考えられる。

そのほかには、学習率や割引率ネットワークの更新回数などのハイパーパラメータの様々な組み合わせをできるだけ多く試し、学習が最もうまくいくパラメータの組み合わせを探ることも必要だと考えられる。また、DQN はその性質上学習のたびに勝率にバラつきが出るため、何回か行った学習結果の平均をとって評価することが望ましいと考えられる。

また、ϵ-greedy 法を用いて今回の実験は行ったが、探索と活用のバランスをどの程度にするかは判断がむずかしいので、最初は ϵ-greedy 法を用いて ϵ を減らしていき、勝率を見てある程度の勝率になれば ϵ の値は減らしたままでもよく、勝率が上がらないようならまだ探索が足りていない可能性があるので ϵ の値を増やすという手法を取ってみる

ことも勝率の上昇につながる可能性があると考えられる。

第4章　結論

本研究は、DQN を用いてブロックスの AI を作成し、ランダムに手を選択するもの、基本的な方策に基づき手を打つ相手を用意し、前者は勝率 8 割程度、後者は勝率 5 割程度を目指すことだった。

実験では、状態として、15 × 15 の盤面の情報とどのピースをどの向きでどの座標に置くかの行動を与えることで、その Q 値を返す DQN を作成した。15 × 15 の盤面の情報は畳み込みニューラルネットワークで処理し、最終的に行動の情報を入力したニューラルネットと結合する方針を取った。バッチサイズの変更や報酬設定の変更を行い勝率に変化が起きるか確認した。

結果としては、ランダムに手を選択する相手には、勝率 6 割程度、基本的な方策に基づき手を打つ相手には、勝率 2 割程度となった。目標にはどちらも及ばずであった。

しかし、学習過程で勝率が上昇していく様子は見られたため、考察で述べたように学習回数を増やす、状態、報酬の種類を増やす、最適なパラメータを模索する等の方法でさらに勝率を高めることができるはずである。

参考文献

1) 大西　紘史, “4 人でプレイする Blokus の AI プレイヤの強化学習”, https://uec.repo.nii.ac.jp/records/9612, (参照 2025-2-21)
2) 株式会社 NTT データ　グローバルソリューションズ, ”機械学習とは？”, 株式会社 NTT データ　グローバルソリューションズ, https://www.nttdata-gsl.co.jp/related/column/what-is-machine-learning.html, (参照 2025-1-20)
3) 斎藤　康毅, “ゼロから作る Deep Learning 4 —強化学習編”, 株式会社オライリー・ジャパン, 2022 年 4 月 4 日
4) 小高　知宏, “基礎から学ぶ人工知能の教科書”, 株式会社オーム社, 2019 年 9 月 25 日
5) CVML エキスパートガイド, ”バッチ正規化 (Batch Normalization) とその発展型”, CVML エキスパートガイド, https://cvml-expertguide.net/terms/dl/layers/batch-normalization-layer/, (参照 2025-1-30)
6) CVML エキスパートガイド, ”ドロップアウト (Dropout)［ランダム正則化］”, CVML エキスパートガイド, https://cvml-expertguide.net/terms/dl/regularization/dropout/, (参照 2025-1-30)
7) ITmedia, ”損失関数 (Loss function) とは?誤差関数/コスト関数/目的関数との違い”, ITmedia, https://atmarkit.itmedia.co.jp/ait/articles/2104/15/news030.html, (参照 2025-2-4)
8) ITmedia, ”［損失関数／評価関数］平均二乗誤差（MSE：Mean Squared Error）／ RMSE(MSE の平方根) とは？”, ITmedia, https://atmarkit.itmedia.co.jp/ait/articles/2105/24/news019.html, (参照 2025-2-4)
9) codexa, ”ダミー変数 (One-Hot エンコーディング) は？”, codexa, https://www.codexa.net/get_dummies/, (参照 2025-2-10)

謝辞

最後に、本研究を進めるにあたり、ご多忙中にも関わらず多大なご指導をしていただきました出口利憲先生、また共に勉学に励んだ同研究室のメンバーに厚く御礼申し上げます。