2048におけるニューラルネットワークプレイヤの育成報告

近藤 直季[1,a)]　松崎 公紀[1,b)]

概要：現在，ニューラルネットワークは，様々な分野で応用される重要な技術のひとつである．ゲームにおいても AlphaGo などニューラルネットワークを適用した人間より強いプレイヤが出現している．本研究では確率的一人ゲームである「2048」に畳み込みニューラルネットワーク（CNN）を適用する．CNN を適用するにあたり，入力層へのデータの与え方，畳み込み層の層数や畳み込み層のチャネル数，畳み込みフィルタのサイズなど，様々なパラメータの組み合わせを考える必要がある．本論文では試行錯誤の末に至った CNN の構成を提案し，3 種類の教師データセットを用いた教師あり学習を行った結果を示す．実験の結果，最も良いもので平均得点 17,228 点，クリア率 28.98% を達成し，最大で 8196 のタイルに到達した．

キーワード：2048，ニューラルネットワーク，教師あり学習，強化学習

1. はじめに

現在，ニューラルネットワークは，様々な分野で応用される重要な技術のひとつである．ニューラルネットワークの中でも，畳み込みニューラルネットワーク（CNN）による多層モデルが，手書き文字の分類から物体や動物の分類といった用途で主に用いられている [13]．ニューラルネットワークはゲームプログラミング分野においても，チェス，将棋，囲碁などへ応用されて良い結果を得ている．特に，ニューラルネットワークによる評価関数とモンテカルロ木探索を併用した AlphaGo [14] は，囲碁の世界トップ棋士に勝利した．

本研究では確率的一人ゲーム「2048」[1] を対象とする．「2048」はスライド&マージ型のゲームのひとつで，とても多くの人間にプレイされている．また，そのコンピュータプレイヤに関する研究も多く行われている．

2048 のコンピュータプレイヤにおいては，これまで N タプルネットワークに基づく評価関数を強化学習により調整したものにより強いプレイヤが作られている．N タプルネットワークによる評価関数を TD 学習により調整するというアイデアは Szubert と Jaśkowski [8] により最初に示された．Jaśkowski [3] によるコンピュータプレイヤは 1 手 1 秒の制限のもとで平均得点 609,104 点を達成しており，これは著者らの知る限り最も高い平均得点である．

一方，ニューラルネットワークによる 2048 プレイヤは，N タプルネットワークを用いたプレイヤほど強いレベルに到達していない．本研究の元となった Gui ら [2] の研究では，畳み込みニューラル

1 高知工科大学
a) 180328e@ugs.kochi-tech.ac.jp
b) matsuzaki.kiminori@kochi-tech.ac.jp

ネットワークによる評価関数を設計し評価を行った結果，最大タイル 4096（1 度のみ），クリア率約 7%，平均得点約 13,000 点と報告されている．この結果は N タプルネットワークに基づく 2048 プレイヤと比べると極端に弱い．しかし，他のゲームにおけるニューラルネットワークの成功を見ると，ニューラルネットワークによる 2048 プレイヤはより強くなる可能性が十分にある．

本研究では，ゲーム盤面の拡張や畳み込み層を多層化するなどのアイデアを適用することにより，より大規模な CNN を設計し，より強いニューラルネットワークプレイヤの作成を目指す．本論文で提案する CNN に対して，既存の強いプレイヤのプレイログより教師あり学習を行った結果，最も良い結果で最大タイル 8192，クリア率 28.98%，平均得点 17,228 点を達成した．なお，この結果は先読み（探索）を取り入れる前のものである．

本論文は，本章を含めて 9 章から構成される．第 2 章では本研究の対象となる 2048 のルールについて述べる．第 3 章では畳み込みニューラルネットワーク（CNN）について述べる．第 4 章では本研究の元となった Gui ら [2] のニューラルネットワークの構成について述べる．第 5 章では本論文で提案する CNN の構成について述べる．第 6 章では提案 CNN を用いた実験の手順と内容について述べる．第 7 章では実験結果と実験結果を受けての考察を述べる．第 8 章では関連研究を示し，第 9 章で本論文をまとめる．

2. 「2048」のルール

確率的一人ゲーム「2048」[1] は，縦 4 マス横 4 マスの計 16 マスからなる盤面を用いる．初期状態はランダムな位置に 2 または 4 のタイルが 2 個配置された盤面である．ゲームは以下の手順を繰り返す．

(1) プレイヤは上下左右のいずれかの方向を入力する．ただし，盤面に変化が起きない方向は選ぶことができない．
(2) 盤面全体のタイルが入力方向へ移動する．この際，同じ数のタイルが入力方向に沿って並ぶ場合は併合され，2 倍の数のタイル 1 個になり，できたタイルと同じ数が得点として加算される．
(3) 盤面の空白マスがひとつランダムで選ばれ，90% の確率で 2 のタイル，10% の確率で 4 のタイルが配置される．

併合はちょうど 2 つのタイルに対して起こる．3 つのタイルが並ぶ場合は，入力方向の奥側より 2 つのタイルが併合する．4 つのタイルが並ぶ場合は，2 倍の数のタイルが 2 個となる（連鎖的な併合は起きない）．

どの方向を選んでも移動または併合が起きない場合ゲーム終了となる．ゲームタイトルである 2048 のタイルを作ることがゲームクリア条件である．ただし，ゲームクリア後も続行可能であり，ゲーム終了時までの総合点をより大きくすることが次の目的となる．

1024 のタイルが初めてできる際の合計得点は約 10,000 点である．同様に，2048 では約 21,000 点，4096 では約 46,000 点，8192 では約 100,000 点である．

3. 畳み込みニューラルネットワーク（CNN）

3.1 畳み込み処理

図 1 は 4×4 の入力画像に対して 2×2 の畳み込みフィルタを掛けた結果として 3×3 の出力画像を得る流れを示したものである．入力画像左上の赤丸の箇所に畳み込みフィルタが掛かる場合は以下の演算により出力画像左上の赤丸箇所の値を得る．

$$1\times2 + 0\times4 + 0\times3 + 0\times1 = 2$$

同様の演算を入力画像に対して右 1 マスずつずらしフィルタを掛け，右端に到達すると左端に戻り下 1 マスずらしフィルタを掛けて行く．このため，4×4 の入力画像に対して 2×2 の畳み込みフィルタを掛ける場合は 9 箇所取れるため出力画像は 3×3 となる．このように 1 マスずつの間隔で畳み込むことをストライド 1 という．図 2 ではストライド 2 の場合を示している．同じ 2×2 の畳み込みフィルタを掛けるが出力画像は 2×2 となる．

図 3 では畳み込みフィルタを 3×3 にした場合を示している．フィルタサイズが大きくなれば出力

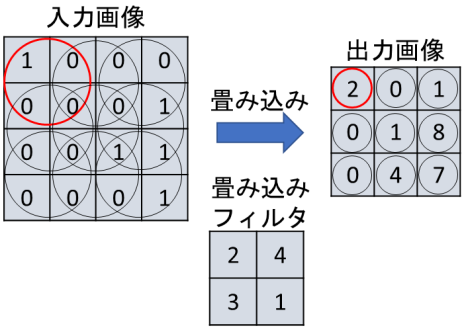


図 1　2×2 フィルタ，ストライド 1 の畳み込み

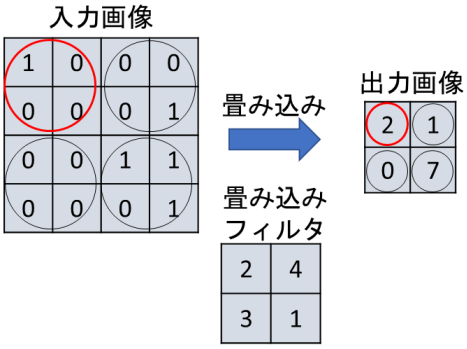


図 2　2×2 フィルタ，ストライド 2 の畳み込み

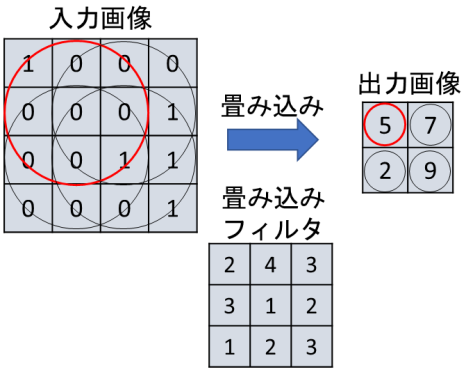


図 3　3×3 フィルタ，ストライド 1 の畳み込み

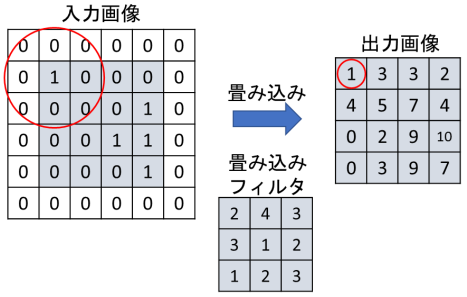


図 4　3×3 フィルタ，ゼロパディングあり，ストライド 1 の畳み込み

画像のサイズは小さくなる．しかし，図 4 のように入力画像の周りを 0 で埋めると出力画像のサイズは入力画像と同じにすることができる．この 0 で埋めることをゼロパディングといい，図 1，2，3 はゼロパディングなし，図 4 はゼロパディングありという．また，ゼロパディングなしでは入力画像の外側を畳み込む回数は内側より少いが，ゼロパディングありでは入力画像の外側を畳み込む回数を増やすことができる．

3.2　全結合層

図 5 は二項分類を想定した CNN における全結合層の例を示すものである．畳み込みの結果得られた 2×2 画像 x の各値 x_j を縦に並べたものが入力に当たる．w_{ij} は重みパラメータで，対応する入力 x_j の重要度が高ければ大きな値を取る．y_i は出力であり，y_1 の求め方は以下の式となる．

$$y_1 = x_1 w_{11} + x_2 w_{12} + x_3 w_{13} + x_4 w_{14}$$

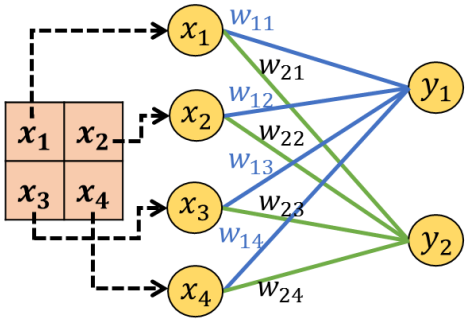


図 5　CNN における全結合層の例

3.3　ソフトマックス関数

図 5 の例では y_1 と y_2 という出力値が得られるが，1（yes）か 0（no）に分類したいため，y_1 と y_2 の値を 0 から 1 の数値（確率）に変換するためにソフトマックス関数を用いる．一般的にソフトマックス関数は以下の式で定義される．P_i は y_i を 0 から 1 の数値に変換したものである．

$$P_i = \frac{e^{y_i}}{\sum_{j=1}^{k} e^{y_j}}$$

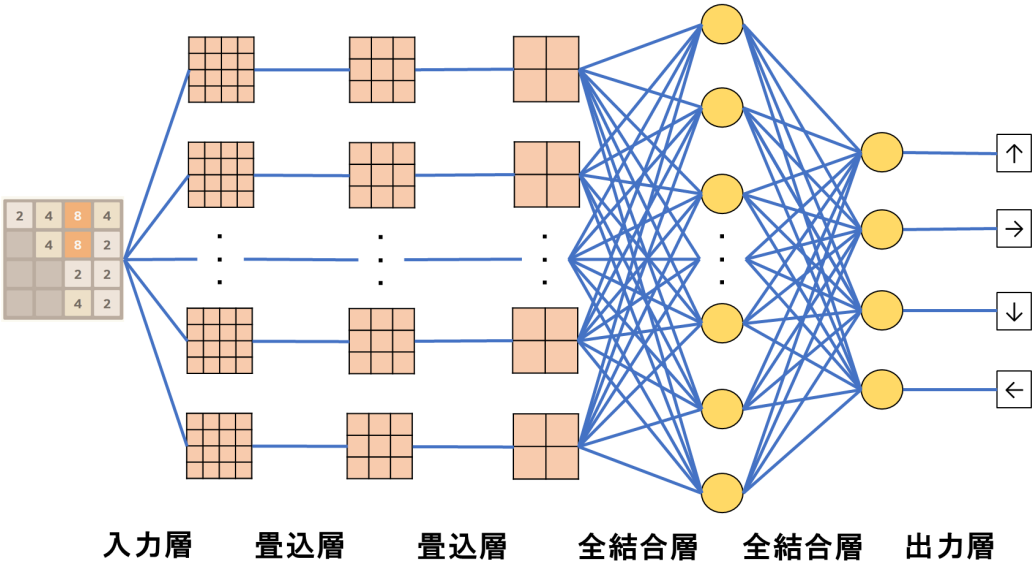


図 6 Gui ら [2] の CNN の全体像

仮に入力画像（犬）を与えた場合に y_1（犬）が 4，y_2（猫）が 3 とすると値の大きい y_1（犬）と判断することができるが正解にどれだけ近いか評価し難い．しかし，ソフトマックス関数によって変換すればこの場合 P_1（犬）は約 0.73 であり，1（正解）にどれだけ近い値か評価することが可能となる．評価ができれば，全結合層の重みパラメータ w を変更（最適化）することでより 1 に近付けること（精度の高い分類）が可能となる．このためソフトマックス関数は CNN における出力層で用いられる．また，CNN において畳み込みフィルタの値も重みパラメータであるため，CNN は畳み込みフィルタと全結合層の重みパラメータを最適化することで精度の高い分類が行えるようになる．

4. Gui らによるニューラルネットワーク

ここでは，Gui ら [2] が用いた CNN について述べる．この CNN はある盤面を入力にとり，その盤面での最善な入力方向を求めるものである．CNN の全体像としては図 6 の通り，入力層，2 層の畳み込み層，2 層の全結合層，出力層からなっている．入力層から全結合層までは空白から 32768 までのタイルに対して独立に計算が行われ，異なる数のタイルの位置関係などは考慮されない．このため，全結合層でようやくそれらの情報を結合することとなる．

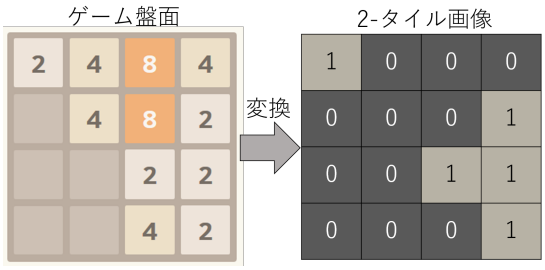


図 7 入力データへの変換（2 のタイルの場合）

4.1 入力層

入力層には 4×4（縦 × 横）の盤面を 16 枚の 4×4 の 2 値画像に変換したデータを与える．16 枚の画像はゲームに登場するタイル，空白，2，4，8，...，8192，16384，32768 のタイルにそれぞれ対応する (65536 のタイルは想定していない)．2 のタイルに対応する画像では，2 のタイルが対応箇所にあれば 1，なければ 0 となる．これを表したものが図 7 である．

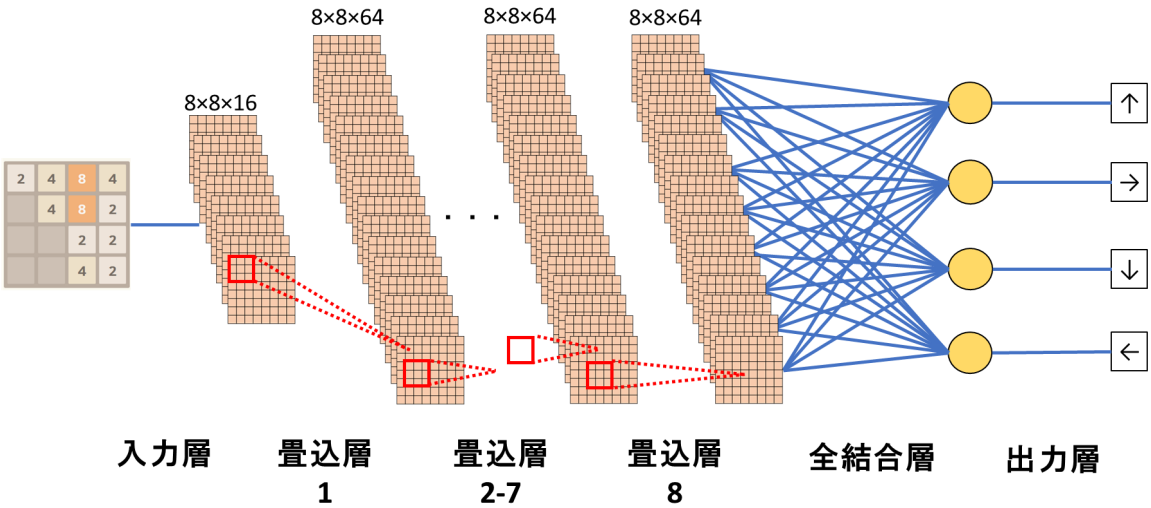


図 8　本研究の CNN の全体像

4.2　畳み込み層

2 層の畳み込み層において，畳み込みフィルタ [*1] のサイズはいずれも 2×2 であり，ゼロパディングなし，ストライド 1 で畳み込む．このため，畳み込み層 1 層目では 4×4 の入力に対して 3×3 の出力になり，2 層目では 3×3 の入力に対して 2×2 の出力となる．加えて，畳み込み層の 1 層目と 2 層目の出力に対して，ReLU 関数を活性化関数として適用する．

4.3　全結合層

全結合層は 2 層からなる [*2]．

4.4　出力層

出力層では上下左右の 4 方向についてソフトマックス関数を用いて値を出力する．

5.　提案手法

ここでは，本研究で用いる CNN の構成について述べる．CNN の全体像としては図 8 の通り，入力層，8 層の畳み込み層，全結合層，出力層からなる．Gui ら [2] のニューラルネットワークとの違いは，入力層へ与えるデータ形式，畳み込み層でのチャネル間結合と層数，フィルタサイズ，畳み込み時のゼロパディングの有無，全結合層の層数である．特に畳み込み層でのチャネル間結合は重要な違いである．

[*1] フィルタ数は明記されていなかった．
[*2] 全結合層の間のノード数は明記されていなかった．

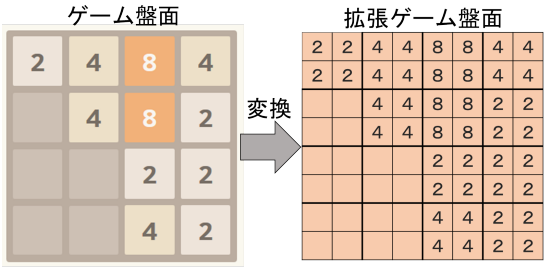


図 9　ゲーム盤面の拡張

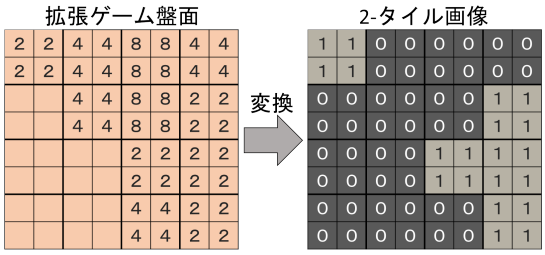


図 10　入力データに変換（2 のタイルのチャネル）

5.1　入力層

まず，4×4 の盤面を図 9 のように 8×8 に拡張する．8×8 に拡張した上で既存手法と同じく図 10 のように，2 値画像にする．本 CNN ではそれらの 16

枚の画像を 16 チャネルの 1 画像として扱う．よって，入力は 8×8×16 となる．

5.2 畳み込み層

畳み込みフィルタのサイズは 8 層とも 3×3 であり，ゼロパディングあり，ストライド 1 で畳み込む．また，各畳み込み層のフィルタ数は 64 とした．このため，8 層ある畳み込み層の出力はそれぞれ 8×8×64 となる．加えて，8 層のすべての出力に対して ReLU 関数を活性化関数として適用する．各畳み込み層は $3\times3\times64 = 576$ 個のパラメータを持つ．

5.3 全結合層

全結合層は 1 層のみで，8×8×64 の値を入力とし，4 個の値を出力する．よって，$4\times(8\times8\times64) = 16384$ 個のパラメータを持つ．

5.4 出力層

全結合層の出力である上下左右の各方向に対して 1 方向に 1 個のバイアス項を加えた後，ソフトマックス関数を用いて値を出力する．

5.5 提案手法に至るまでの経緯（考察）

図 6 の構成に関して著者らは「2^k のタイルを作るためには 2^{k-1} のタイルが 2 つ必要」という 2048 のゲーム特性から，畳み込み層の入力には 2^k のタイルの情報に加えて 1 つ小さい 2^{k-1} のタイルの情報も与えるべきではないかと考えた．このため，あるタイルの 2 チャネル間について重視した CNN を構成したが，平均得点 3000 点程度とランダムプレイヤと同レベルの結果となった．

次に 2^k のタイルの前後に当たる 2^{k-1} と 2^{k+1} のタイルの情報を与えるという，3 チャネル間について重視した CNN を構成した．しかし，2 チャネル間よりわずかに良いがランダムプレイとヤ同レベルのままであった．

これらの試行錯誤の中でフィルタ数を 2, 4, 8, ..., 1024 と増やすほど平均得点が向上することを確認できた．しかし，フィルタ数がある一定値（CNN の構成により異なる）を超えると平均得点が低くなることがわかった．また，これらのクリア率は 1%未満であった．

アルファ碁の仕組みに関する書籍 [14] によると，アルファ碁が勝率予測で用いる CNN の畳み込み層は 13 層から成り，ゼロパディングありで畳み込む構成であった．著者らはこれまでゼロパディングなしで構成していたため，畳み込み層は 3 層以下でしか構成できなかった（4×4 画像を 2×2 フィルタで 3 回畳み込むと 1×1 画像になるため）．このため，ゲーム盤面を 8×8 に拡張することにより畳み込み層の層数を増やせるのではないかというアイデアに至った．また，ゼロパディングなしで畳み込むことで盤面を小さくし，情報量を減らすことに意味があると考えていたが，アルファ碁に習いゼロパディングありとした．

その後，入力層へ与えるデータ形式を変更し，畳み込み層の層数を 4，8，12 層の 3 つを簡易に検証したところ 4 層より 8 層と 12 層が良い結果を出し，8 層と 12 層では大差なかったため，学習時間の短い 8 層での提案に至った．畳み込み層の層数および各層のフィルタ数は検証の余地がある．

6. 実験手順

学習には，1 回の学習で 1000 盤面ずつ与えるバッチ学習を用いる．また，学習の途中経過を教師データセットの 5% 毎，つまり 5×10^6 盤面の学習が完了する毎に評価する．評価事項はテストデータセットを用いた選択した手の一致率，2048 を 1 万ゲームプレイさせたときの平均得点，最大得点，クリア率とした．多くの 2048 プレイヤは expectmax 探索といった探索による先読みを行っているが，この CNN では先読みを行っていない．

6.1 教師データセットとテストデータセット

盤面とその盤面での入力方向を対にしたものを，1 局面の教師データとする．教師データセットとして，以下の 3 種類のものをそれぞれ 10^8 局面分用意した．これらのプレイヤの詳細については文献 [5] を参照のこと．

TR30　8 ステージからなる N タプルネットワークを，強化学習（Temporal Coherence 学習）により調整し，3 手先読みの Expectimax 探索によりプレイしたもの．平均得点は約 30 万点であり，約 8000 ゲームからなる．

TR45　8 ステージからなる N タプルネットワークを，強化学習（Temporal Coherence 学習）により調整し，リスタート方策ありの 3 手先読みの Expectimax 探索によりプレイしたもの．平均得点は約 45 万点であり，約 6000 ゲームからなる．

TR50　TR45 と同じ N タプルネットワークを 4 手先読みの Expectimax 探索によりプレイしたもの．平均得点は約 50 万点であり，約 5500 ゲームからなる．

テストデータセットは，TR50 と同じプレイヤのプレイログとした．教師データセットと異なる 100 ゲーム分のプレイログより 10^4 局面をランダムに選択したものである．

6.2　教師あり学習によるパラメータの最適化

提案手法の CNN では，ある盤面に対して最善の方向を返すように設計した．その教師あり学習では，教師データと同じ方向を選ぶ確率 P を最大化するよう学習を行う．

最適化の目標として，確率 P を最大化することと交差エントロピーを最小化することは同じであるため，交差エントロピーを用いた次式が誤差関数 E である．t は教師データの正解ラベルで 4 方向に対して 0 か 1 の値を取り，前者は不正解，後者は正解の意味を持つ．

$$E = -\sum_{i=0}^{3} t(i) \times \log P(i)$$

最適化は誤差関数 E を定義したうえで TensorFlow のトレーニングアルゴリズムのひとつ `tf.train.AdamOptimizer` によって行った．この際の学習率はデフォルトの 0.001 とした [13]．また，畳み込みフィルタのパラメータと全結合層のパラメータの初期値は −0.1 から 0.1 までの値をランダムで与えている．出力層のバイアス項の初期値は 0 とした．

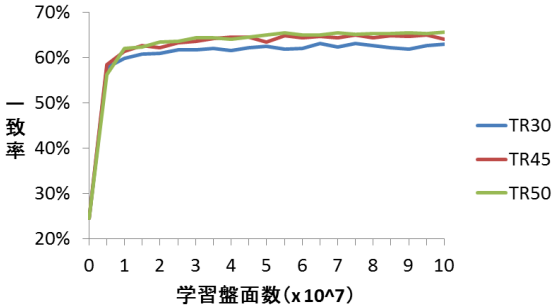


図 11　学習段階における一致率の推移

7.　実験

7.1　実験結果

テストデータセットとの選択した手の一致率を図 11 に示す．教師データを全く与えていない図の左端では 3 種類とも一致率はおよそ 25% であり，これは 4 方向の入力しかない 2048 におけるチャンスレベルと同等である．3 種類ともに 5×10^6 から 10^7 局面の学習が完了した段階で一致率の落ち着きが見られる．10^8 局面学習後の一致率については，TR30 では 62.88%，TR45 では 63.97%，TR50 では 65.54% である．TR50，TR45，TR30 の順に一致率が高い．

教師データを 5×10^6 局面学習する毎に，その学習段階で 2048 をプレイさせ，平均得点をまとめたものが図 12 である．10^8 局面学習後の平均得点については，TR30 では 16,643 点，TR45 では 10,706 点，TR50 では 7,694 点である．また，最も平均得点が高い箇所については，TR30 では 8.5×10^7 局面学習後の 17,228 点，TR45 では 9×10^7 局面学習後の 13,215 点，TR50 では 8.5×10^7 局面学習の 8,843 点である．一致率の高さとは逆に TR30，TR45，TR50 の順に平均得点が高い．

同様に，最大得点をまとめたものが図 13 である．この最大得点とは各学習段階での 1 万ゲームプレイした中で最も大きい 1 ゲームの得点を指す．10^8 局面学習後の最大得点については，TR30 は 103,224 点，TR45 は 96,448 点，TR50 は 78,388 点である．また，最も最大得点が高い箇所については，TR30 は 10^8 局面学習後の 103,224 点，TR45 は 9×10^7 局面学習後の 123,624 点，TR50 は 9.5×10^7 局面

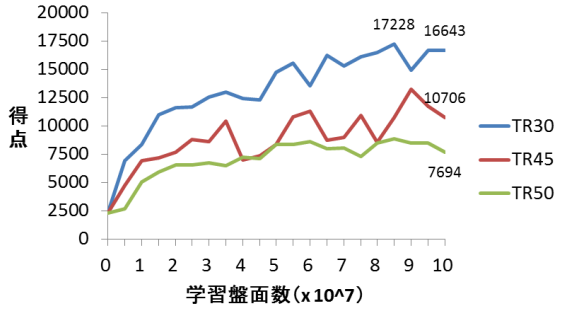


図 12　学習段階における平均得点の推移

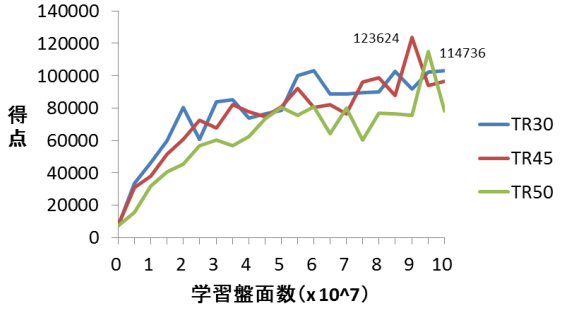


図 13　学習段階における最大得点の推移

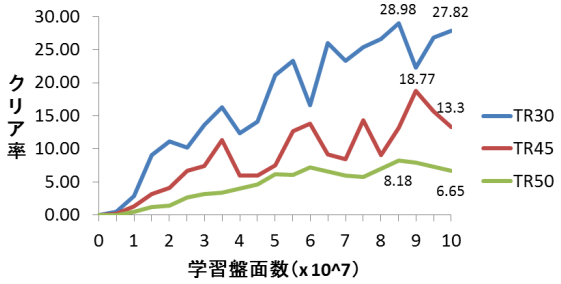


図 14　学習段階におけるクリア率の推移

学習後の 114,736 点であり，全て 8192 のタイルができている．

同様に，クリア率をまとめたものが図 14 である．10^8 局面学習後のクリア率については，TR30 は 27.82%，TR45 は 13.3%，TR50 は 6.65% である．また，最もクリア率が高い箇所については，TR30 は 8.5×10^7 局面学習後の 28.98%，TR45 は 9×10^7 局面学習後の 18.77%，TR50 は 8.5×10^7 局面学習後の 8.18% である．

次に，10^8 局面学習後に 1 万ゲームプレイさせた際の各ゲームの最大タイルについてまとめたものが TR30，TR45，TR50 の順に図 15，16，17 である．この最大タイルとは，1 ゲームのゲーム終了時に盤面にある一番値の大きなタイルを指す．例えば，ある 1 ゲームでゲーム終了時の盤面にある一番値の大きなタイルが 2048 の場合は最大タイルは 2048 である．

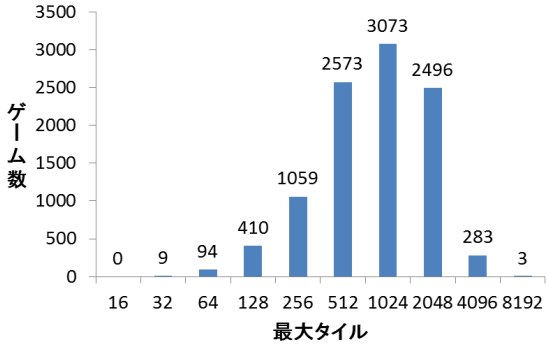


図 15　TR30 の最大タイル分布

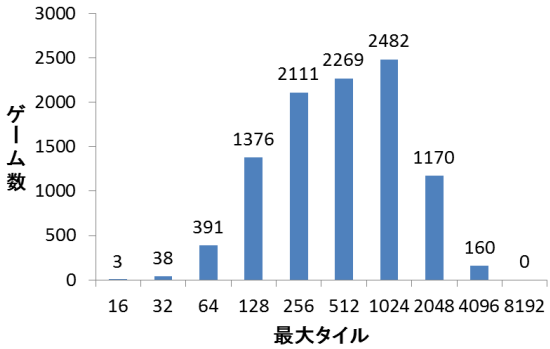


図 16　TR45 の最大タイル分布

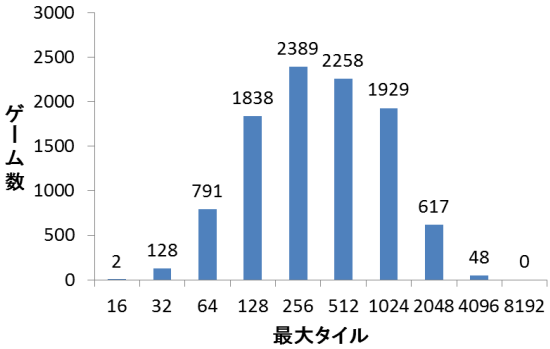


図 17　TR50 の最大タイル分布

7.2　考察

初めに，教師データセット TR50 を学習させた

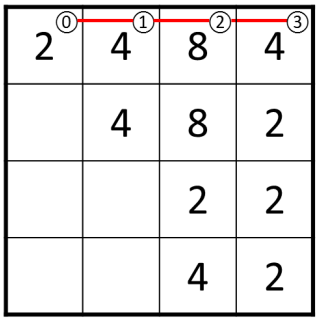
CNN プレイヤが最も一致率が高いがクリア率が低いことに関して，教師データセット TR50 とテストデータセットのプレイログを生成したプレイヤは同じであることが，一致率が他より高い要因の可能性がある．このため，別プレイヤのプレイログを用いた評価も行うべきである．

次に，実験前には，与えた教師データセットの元となるプレイヤの強い順，データセット名としては TR50，TR45，TR30 の順に結果が良くなることを想定していた．しかし前節の通り，教師データセット TR30 を学習させた CNN を用いたプレイヤが一番良い結果を出した．これは，本実験前まで教師データセットは TR30 のみを用いて最適な CNN を模索してきた経緯があり，提案した CNN の重要な構成要素である畳み込み層の層数や畳み込みフィルタの数などが TR30 に対しては良く適したが，TR45 と TR50 に対しては良く適さなかったことが理由のひとつかもしれない．他には，学習局面数が増える度に平均得点，クリア率は増加しているため 10^8 局面の学習量では足りていない可能性が考えられる．

最後に最大タイルの分布に関して，図 15，16，17 の通り，最大タイルが 16，32，64，128 のゲーム数が少なくとも 5% 以上存在する．（第一著者を含む）多くのプレイヤにとって，最大タイルが 128 以下というのはまずありえない悲惨な結果である．当然そのようなプレイが多くあれば平均得点の低下に繋がる．提案した CNN では現盤面の判断しかできないため，新たに評価関数を用意することで 3 手程の先読みの探索を行うことが必要だと考える．

8. 関連研究

「2048」に対するコンピュータプレイヤにおいて，N タプルネットワークと強化学習を組み合わせる手法が有用であることが様々な研究により示されている．そのアイデアは，Szubert と Jaśkowski [8] により最初に提案された．ゲーム盤面の N 個のマスの組を N タプルという．図 18 の場合はゲーム盤面の上段 4 マスの組からなる 4 タプルの一例で，対応するタイルの 2 の指数を取って（空きマスは 0 と対応付ける）テーブルを引く．よって「2-4-8-4」の場合は「1-2-3-2」に対応する評価値「7.9」を得る．図 19 は 4 つの 6 タプルであり，この場合はあるゲーム盤面において，4 つの 6 タプルの評価値の合計値によりプレイヤの行動を選択する．このように複数の N タプルを組み合わせる手法が N タプルネットワークである．その後 Wu ら [11] により，N タプルネットワークがマルチステージ化を用いて拡張された．Yeh ら [12] は，このアイデアに基づき，3 ステージの N タプルネットワークを評価関数した場合，貪欲プレイにより平均得点 143,958 点クリア率 96.6%，3 手先読みの Expectimax 探索により平均得点 350,394 点クリア率 99.9% を達成した．また，より多くのステージと人手で設計した評価項目とを併用した場合，3 手先読みの Expectimax 探索にて平均得点 443,526 点を達成した．著者らが知る限りの最も良いコンピュータプレイヤは，Jaśkowski によるもの [3] である．そこでは，Temporal Coherence 学習，マルチステージ化，冗長な N タプルなどの手法が適用され，3 手先読みの Expectimax 探索により平均得点 511,759 点，1 手 1 秒の制限時間のもとで平均得点 609,104 点を達成した．本研究で用いた教師データは，Matsuzaki [5] による N タプルネットワークと強化学習（リスタート方策ありの



⓪	①	②	③	評価値
0	0	0	0	-1.3
0	0	0	1	2.1
⋮	⋮	⋮	⋮	⋮
0	0	1	0	-3.4
⋮	⋮	⋮	⋮	⋮
1	**2**	**3**	**2**	**7.9**
⋮	⋮	⋮	⋮	⋮

図 18 4 タプルに基づく評価関数の例

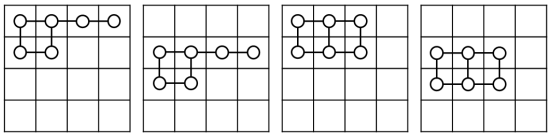


図 19 Wu らが用いた 4 つの 6 タプル [11]

Temporal Coherence 学習）によるプレイヤにより生成した．これらの研究とは独立して，評価関数として用いる N タプルの系統的な選択法について，Oka と Matsuzaki [4], [6] による研究がある．

一方，「2048」に対して，多層ニューラルネットワークを用いた評価関数を用いることは，これまであまり研究されていない．著者らが知る限り，論文として報告されているものは Gui によるもの [2] だけである．論文として報告はされていないが，オープンソースとして公開されているプログラムには，ニューラルネットワークに基づくもの [9]，深層 Q 学習を用いるもの [10]，深層リカレントニューラルネットワークによる Q 学習を用いるもの [7] などがある．このうち，プログラム [9] は，クリア率 94.152% を達成したと報告している．

9. まとめ

確率的一人ゲームである「2048」を対象とした畳み込みニューラルネットワークの構成を提案した後，3 種類の教師データセットを用いて教師あり学習を行った．結果としては 8192 のタイルに到達することができ，最も良いものでクリア率 28.98%，平均得点は 17,228 点となった．これは Gui ら [2] の最大タイル 4096，クリア率約 7%，平均得点約 13,000 点という結果を上回る．しかしながら，N タプルネットワークに基づくプレイヤ [3] の結果（貪欲プレイで平均得点 324,710 点，探索ありで平均得点 609,104 点）には程遠い結果であった．

また，最大で 8192 のタイルに到達する反面，256 以下で終了するゲーム数が少なくとも 1 割を超えており，平均得点に大きく影響を与えている．この影響の解決法として提案した CNN に数手先を読む探索を実装することが今後の課題である．

謝辞　本研究の実験で用いた教師データは，高知工科大学の IACP クラスタ計算機を使用して得られたものである．

参考文献

[1] Cirulli, G.: 2048, `http://gabrielecirulli.github.io/2048/` (2014).

[2] Gui, H., Wei, T., Huang, J.-B. and Wu, I.-C.: An Early Attempt at Applying Deep Reinforcement Learning to the Game 2048, *Workshop on Neural Networks in Games* (2016).

[3] Jaśkowski, W.: Mastering 2048 with Delayed Temporal Coherence Learning, Multi-Stage Weight Promotion, Redundant Encoding and Carousel Shaping, *IEEE Transactions on Computational Intelligence and AI in Games* (2017).

[4] Matsuzaki, K.: Systematic Selection of N-tuple Networks with Consideration of Interinfluence for Game 2048, *Proceedings of the 2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI 2016)* (2016).

[5] Matsuzaki, K.: Developing 2048 Player with Backward Temporal Coherence Learning and Restart, *Proceedings of Fifteenth International Conference on Advances in Computer Games (ACG2017)* (2017).

[6] Oka, K. and Matsuzaki, K.: Systematic Selection of N-tuple Networks for 2048, *Proceedings of 9th International Conference on Computers and Games (CG2016)*, Vol. LNCS 10068, Springer, pp. 81–92 (2016).

[7] Samir, M.: 2048 Deep Recurrent Reinforcement Learning. `https://github.com/georgwiese/2048-rl`.

[8] Szubert, M. and Jaśkowski, W.: Temporal Difference Learning of N-Tuple Networks for the Game 2048, *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–8 (2014).

[9] tjwei: A Deep Learning AI for 2048. `https://github.com/tjwei/2048-NN`.

[10] Wiese, G.: 2048 Reinforcement Learning. `https://github.com/georgwiese/2048-rl`.

[11] Wu, I.-C., Yeh, K.-H., Liang, C.-C., Chang, C.-C. and Chiang, H.: Multi-Stage Temporal Difference Learning for 2048, *Technologies and Applications of Artificial Intelligence*, Lecture Notes in Computer Science, Vol. 8916, pp. 366–378 (2014).

[12] Yeh, K.-H., Wu, I.-C., Hsueh, C.-H., Chang, C.-C., Liang, C.-C. and Chiang, H.: Multi-stage temporal difference learning for 2048-like games, *IEEE Transactions on Computational Intelligence and AI in Games* (2016).

[13] 中井悦司：TensorFlow で学ぶディープラーニング入門–畳み込みニューラルネットワーク徹底解説–，株式会社マイナビ出版 (2016).

[14] 大槻知史：最強囲碁 AI アルファ碁 解体新書深層学習、モンテカルロ木探索、強化学習から見たその仕組み，株式会社翔泳社 (2017).