令和元年度

修士学位論文

2048 における盤面の対称性を考慮した CNN プレイヤの学習

Training CNN Players Considering Board Symmetry in Game 2048

1225119　　近藤 直季

指導教員　　松崎 公紀

2020 年 2 月 28 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報学コース

要 旨

2048 における盤面の対称性を考慮した CNN プレイヤの学習

近藤 直季

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 AlphaZero 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」に畳込みニューラルネットワーク（CNN）を適用する．2 つの提案する CNN 構成において，教師データセットを用いた教師あり学習を行った.「2048」における対称性を CNN の学習後のプレイ方法に取り込む方式（提案 1）と畳込み層では独立に特徴抽出を行うが全結合層で結合することで学習時から対称性を考慮する方式（提案 2）を調査した．

提案 1 では畳込み層を 2 層から 9 層の 8 種類のモデルと 5 層において異なるチャネル数を持つ 4 種類のモデルを検証し，提案 2 では畳込み層を 3 層から 6 層に絞り，層数とチャネル数が異なる 10 種類のモデルを検証した．

提案 1 では最も良いモデルで平均得点 118,434 点，最高得点 386,812 点，クリア率 89.5%を達成し，提案 2 では最も良いモデルで平均得点 119,637 点，最高得点 533,624 点（32768 のタイルに到達），クリア率 80.7%を達成した．

提案 2 の最も良いモデルにおいてテストプレイを行った際，提案 1 と比べ序盤（512 のタイル未満）でゲームオーバーになる確率は 10.4%高く，16384 のタイル以上の到達率は提案 1 より 5.1%高いことがわかった．序盤でのゲームオーバーは平均得点を著しく低下させる要因であるため，改善が今後の課題である．

キーワード　2048，ニューラルネットワーク，教師あり学習

Abstract

Training CNN Players Considering Board Symmetry in Game 2048

Naoki Kondo

Currently, neural networks are regarded as important technologies applied in various fields. In game informatics, neural networks are used to develop players that are stronger than humans, such as AlphaZero. In this study, convolutional neural network (CNN) are applied to stochastic single-player game "2048". We proposed two configurations of CNNs, for which supervised learning was performed with a training dataset. In the first method (Proposal 1), we train a CNN in a simple way and incorporate the symmetry in "2048" in the play method. In the second method (Proposal 2), we consider the symmetry in "2048" in the training of CNN that combines the symmetric information at the fully-connected layer. In Proposal 1, eight models with two to nine convolutional layers and four models with different number of channels in five layers were investigated. In Proposal 2, ten models with different numbers of layers and different numbers of channels were verified. In Proposal 1, the best model achieved an average score of 118,434, a maximum score of 386,812, and a clearing rate of 89.5%. In Proposal 2, the best model achieved an average score of 119,637, a maximum score of 533,624 (reaching a 32768 tiles), and a clear rate of 80.7%. When test-playing with the best model in Proposal 2, the probability of early game over (less than 512 tiles) was 10.4% higher than in Proposal 1, and that the reaching 16384 tiles was 5.1% higher than in Proposal 1. Since early game over significantly reduces the average score, its

improvement remains as a future issue.

key words 2048, neural network, supervised learning

目次

図目次

表目次

第 1 章

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用いられている [1]．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，良い結果が報告されている．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 AlphaZero[2] は，チェス，将棋，囲碁において人間のトップ棋士に勝利した各コンピュータプレイヤに対して勝ち越した．

本研究では確率的一人ゲーム「2048」[3] 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，単純なルールであるが難しいという特徴は多くの人々を魅了している．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 [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]．

「2048」のコンピュータプレイヤにおいては，N-tuple 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で強いプレイヤが作れらている．N-tuple 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 [10] により最初に示された．Jaśkowski [5] 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-tuple 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 Guei ら [4] の研究では，畳込みニューラル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，平均得点約 11,400 点，クリア率約 7%，最高で 4096 のタイルに 1 度だけ到達したと報告されている．この結

果は N-tuple 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，畳込み層を多層化することや重みパラメータ数を増やすことにより，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．本論文では 2 つの CNN 構成を提案する．提案 1 は作成した CNN プレイヤがランダムプレイを超えた段階から改良を重ねて行き着いたものであり，提案 2 は提案 1 の改良途中で判明した「2048」における対称な 8 盤面の情報を CNN 内で完結させたものである．

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，提案 1 では最も良い結果で平均得点 118,434 点，最高得点 386,812 点，クリア率 89.5%，を達成し，提案 2 では最も良い結果で平均得点 119,637 点，最高得点 533,624 点（32768 のタイルに到達），クリア率 80.7%を達成した．なお，この結果は先読み（探索）を行っていない．

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」のルールと対称性について述べる．第 3 章では畳込みニューラルネットワーク（CNN）についてと，本研究の元となった Guie ら [4] の CNN の構成について述べる．第 4 章では提案 1 の CNN 構成について述べる．第 5 章では提案 2 の CNN 構成について述べる．第 6 章では提案 1 の CNN 構成で実験を行った結果を述べる．第 7 章では提案 2 の CNN 構成で実験を行った結果を述べる．第 8 章では提案 1 と提案 2 の実験結果の比較とその考察を述べる．第 9 章で本論文をまとめる．

第 2 章

ゲーム「2048」

「2048」は G.Cirulli によって 2014 年に公開された 1 人用のパズルゲームである [3]．本章では,「2048」のルールを述べた後，本研究において重要な要素である「2048」における対称性について述べる．

2.1 「2048」のルール

「2048」は，縦横 4×4 マスの盤面で行われ，空白と「2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536」の計 16 種類のタイルが存在する．初期盤面は 16 マスの空白マスのうちランダムな 2 箇所の異なるマスに 2 または 4 のタイルが出現した状態である．ゲームは次の流れを繰り返す．本家には手数という概念がないが便宜上これを 1 手と数える．

1. プレイヤは上下左右のうち任意の 1 方向を入力する．ただし，盤面に変化が起きない場合は無効となる．
2. ゲーム内処理として，入力された方向に向かってタイルが端まで移動する．移動の際，同じ数値のタイルが入力方向に沿って並ぶ場合は端から順に 2 つが併合され，2 倍の数値のタイルが 1 つになり，できたタイルと同じ数値が得点として加算される．
3. 盤面の空白マスに対してランダムで 1 箇所が選ばれ，90%の確率で 2 のタイル，10%の確率で 4 のタイルが出現する．

併合に関して，例えば左に入力された場合,「2, 2, 空白, 空白」は「4, 空白, 空白, 空白」となる.「2, 2, 2, 空白」は「4, 2, 空白, 空白」となり,「2, 2, 2, 2」は「4, 4, 空白, 空白」となる.

どの方向に入力しても，移動または併合が起きない場合はゲーム終了となる．ゲームのタイトルである「2048」のタイルを作ることがゲームクリアの条件となる．また，クリア後も続行可能であり，ゲーム終了時までに積み重ねた得点の合計を大きくすることが次の目標となる.

よって,「2048」のコンピュータプレイヤにおける強さの指標は，ゲーム終了時の合計得点の平均（以降，平均得点とする），合計得点の最高点（以降，最高得点とする），2048 のタイルに到達できた割合（以降，クリア率とする），到達できた最大のタイル，生き延びた手数の平均などが挙げられる.

2.2 「2048」における対称性

「2048」において，ある 1 つ盤面は回転と反転によって図 2.1 のように合計 8 個の盤面を得ることができる．また，本研究において回転は時計回りであり，反転は左右反転を指す.

2.2 「2048」における対称性

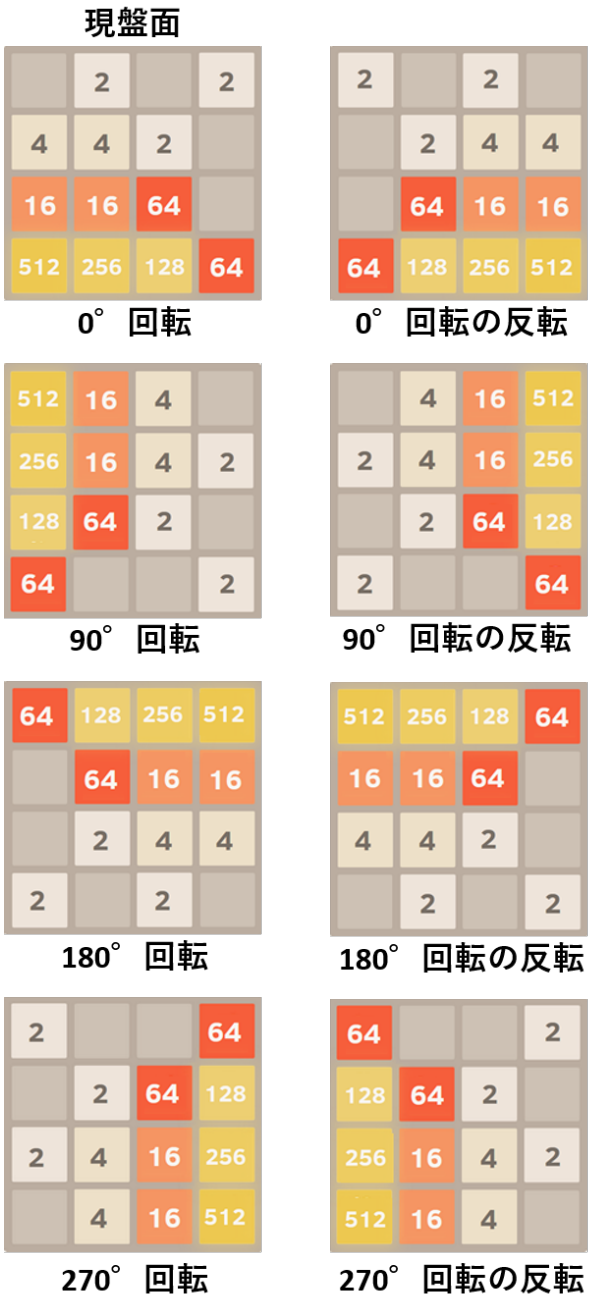


図 2.1　2048 における対称性の例

第 3 章

畳込みニューラルネットワークについて

ニューラルネットワークにおける中間層で畳込みフィルタを用いた畳込み処理を行うものを畳込みニューラルネットワーク（CNN : Convolutional Neural Network）という．本章では，CNN に関する要素技術について述べた後，先行研究である Guei らの CNN の構成とアイデアについて述べ，本研究における 2 つの CNN 構成（提案 1，提案 2）について述べる．

3.1 要素技術

3.1.1 畳込み処理

図 3.1 は 4×4 の入力画像に対して 2×2 の畳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込みフィルタを掛けると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込むことをストライド 1 という．図 3.2 ではストライド 2 の場合を示して

いる．同じ 2×2 の畳込みフィルタを掛けるが出力画像は 2×2 となる．

図 3.3 では畳込みフィルタを 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画像のサイズは小さくなる．しかし，図 3.4 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 3.1，3.2，3.3 はゼロパディングなし，図 3.4 はゼロパディングありという．

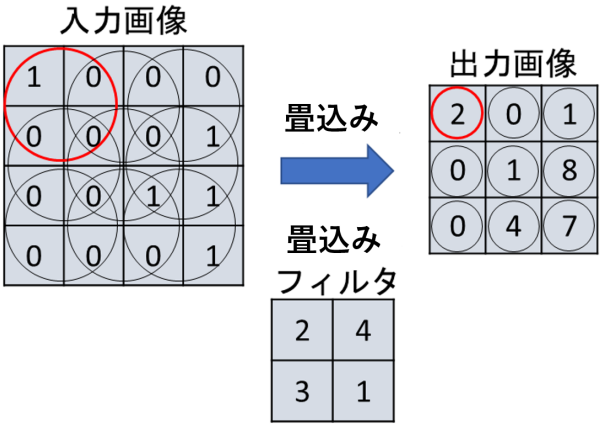


図 3.1　2×2 フィルタ，ストライド 1 の畳込み

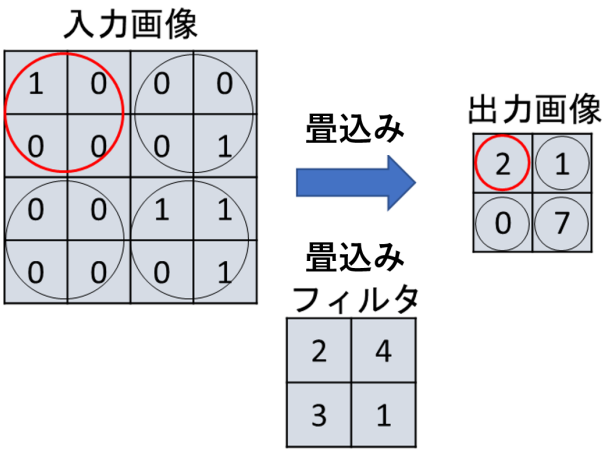


図 3.2　2×2 フィルタ，ストライド 2 の畳込み

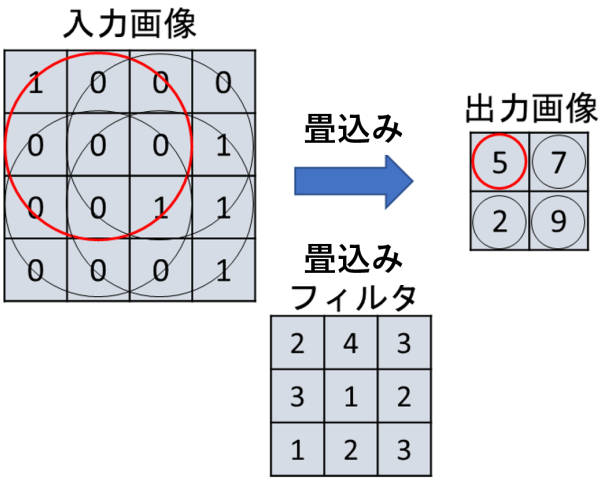


図 3.3　3×3 フィルタ，ストライド 1 の畳込み

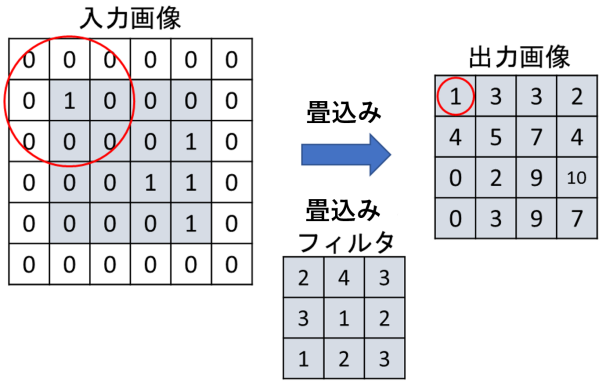


図 3.4　3×3 フィルタ，ゼロパディングあり，ストライド 1 の畳込み

3.1.2　全結合層

図 3.5 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．畳込みの結果得られた 2×2 画像の各値 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．

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

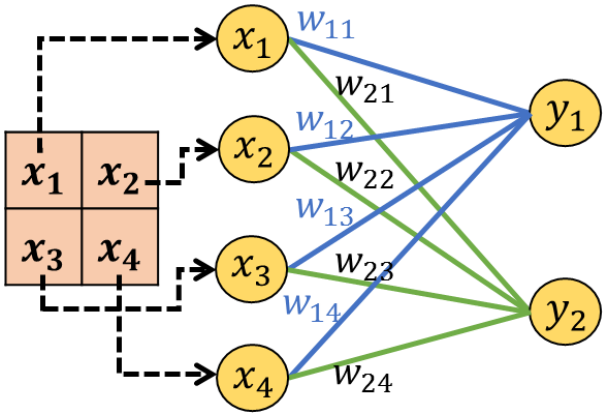


図 3.5　CNN における全結合層の例

3.1.3　ソフトマックス関数

図 3.5 の例では y_1 と y_2 という出力値が得られた．0（分類 A）か 1（分類 B）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が用いられている．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．また P の総和は 1 となる．

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

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ている．

3.2　Guei らの CNN の構成とアイデア

ここでは，先行研究にあたる Guei ら [4] が用いた CNN について述べる．CNN の構成は入力層，2 層の畳込み層，2 層の全結合層，出力層からなっている．本研究における入力層のデザインは Guie らのアイデアを用いてるため掘り下げる．全結合層と出力層に関する詳細は記載されていないため説明を省く．

学習方法は強化学習を用いていた．また，数値は明記されていないが，グラフを読み取ると，平均得点は約 11,400 点，最高得点は約 50,000 点，クリア率は約 7%である．

3.2.1　入力層

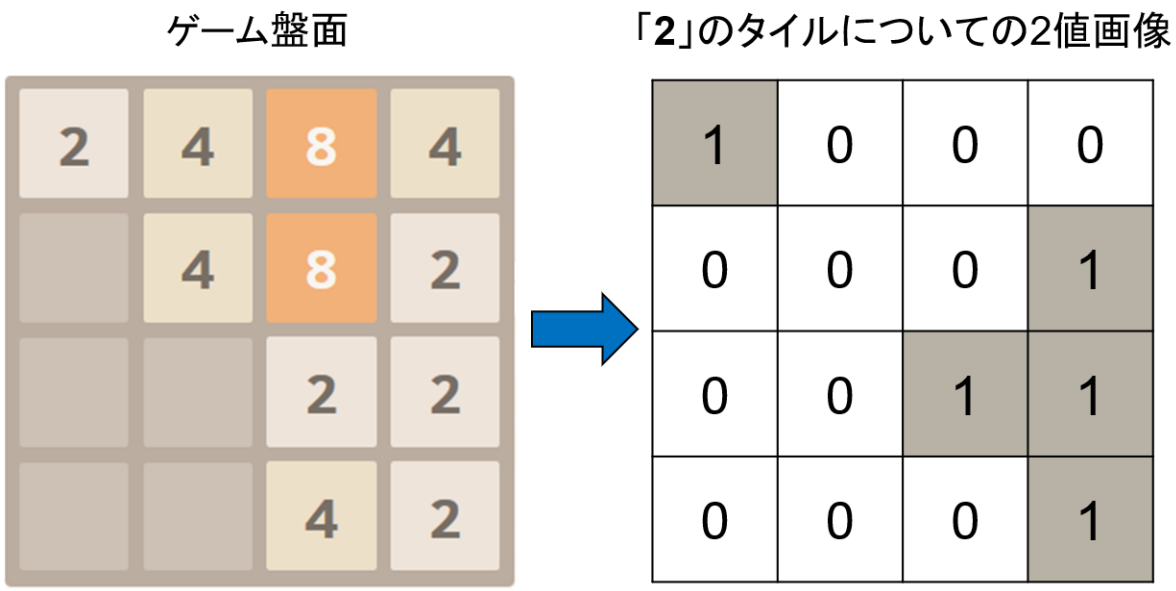


図 3.6　「2」タイルを 2 値画像へ変換

入力層には 4×4（縦 × 横）のゲーム盤面を 16 枚の 4×4 の 2 値画像に変換したデータを

与える．16 枚というのは 2048 に登場する「空白, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 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」のタイルがゲーム盤面にあれば 1，なければ 0 とする．これを表したものが図 3.6 である．

3.2.2　畳込み層

2 層の畳込み層において，畳込みフィルタのサイズはどちらも 2×2 であり，ゼロパディングなし，ストライド 1 で畳込む．また，各畳込み層の出力には ReLU 関数を活性化関数として適用している．このため，畳込み層 1 層目の入力は 4×4 に対して，出力は 3×3 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 (Ch 数は記載されていないため省略している).

従って，この CNN 構成では畳込み層を 3 層にすると 3 層目の出力は 1×1 となり，フィルタサイズを 2×2 のままで畳込み層を 4 層以上にすることはできない.

第 4 章

提案 1：対称性を考慮しない CNN

画像認識のコンテストである ILSRVRC (ImageNet large scale visual recognition challenge) では 2012 年以降は CNN を用いたものが優れた結果を残している [15]．年々性能が向上しており，その背景には中間層の増加が 1 つの要因であると考えた．

このため本研究では，中間層である畳込み層を増加させる．Guei らの CNN 構成では，ゼロパディングなしとしてあるため，2×2 の畳込みフィルタで畳込める回数に限界があった．そこで，本研究ではゼロパディングありとすることで，ゲーム盤面と同じ 4×4 のサイズを維持していくらでも畳込み層を増やせるようにした．

畳込み層は 2 層から 9 層の 7 通りを作成し，全結合層は 1 層としている．提案 1 の CNN の全体像 (畳込み層が 4 層の場合) は図 4.1 となっている．また，すべての畳込み層に対して同じ数 (k) のフィルタ数を使用する．このため，図 4.1 の場合は k =256 のものを示している．各層に関する説明は次の通りである．

4.1　入力層

入力層は Guei らの方法と同じく 4×4 の Ch 数 16 とした．

4.2　畳込み層

畳込み層は，フィルタサイズ 2×2 を k 枚として，右側と下側にゼロパディングを行い，入力を 4×4 から 5×5 にした後，ストライド 1 で畳込む．畳込みの結果得られる 4×4 の k Ch に対して 4×4 の k Ch のバイアス項（初期値 0.1）を加えた後，ReLU 関数を適用した

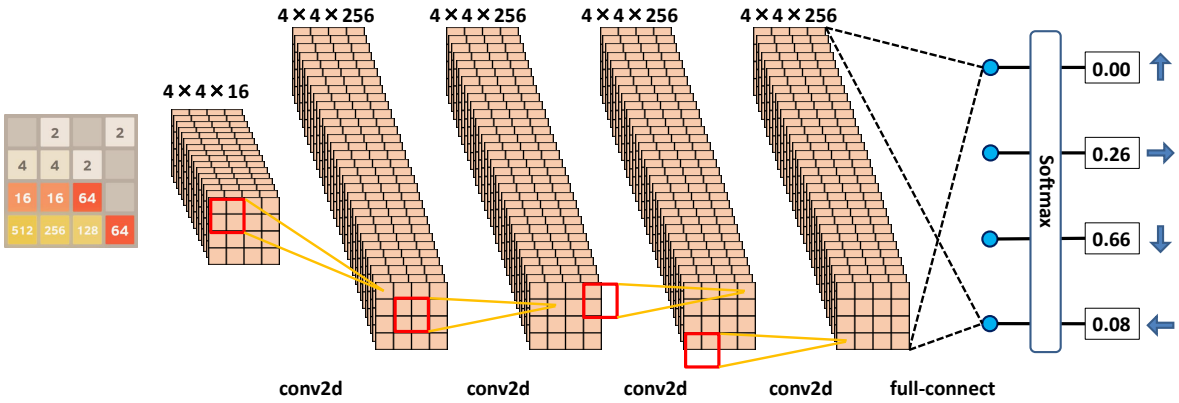


図 4.1　提案 1 の CNN の概要図

ものを畳込み層 1 層における出力とする．また，畳込みフィルタの初期値は TensorFlow の関数で「`truncated_normal(stddev=0.1)`」とした．

4.3　全結合層

畳込み層の出力である 4×4 の k Ch を 1 次元化した $4\times4\times k=16k$ を入力に与える．出力はゲーム盤面で入力すべき 4 方向のそれぞれの値とするため 4 で，これにバイアス項を加えたものである．

4.4　出力層

全結合層の出力である 4 つの値をソフトマックス関数により上右下左の各方向に対する確率に変換する．

4.5　パラメータ数の算出方法

提案 1 の CNN におけるパラメータ数の算出は以下の通りである．

畳込み層 1 層目はフィルタサイズ 2×2 を Ch 数 16 に対して k 枚のフィルタを掛けるため

$$1\text{ 層目} = (2 \times 2 \times 16) \times Ch(k)$$

畳込み層 2〜n 層目はフィルタサイズ 2×2 を Ch 数 k に対して k 枚のフィルタを掛けるため

$$2\text{ 層目}〜n\text{ 層目} = (2 \times 2 \times Ch(k)) \times Ch(k) \times (n-1)$$

全結合層は 1 次元化して 4 つの出力とするため

$$\text{全結合層} = (4 \times 4 \times Ch(k)) \times 4$$

バイアス項は全結合層で 4，畳込み層では 1Ch につき 1 つを n 層のため

$$\text{バイアス項} = 4 + Ch(k) \times n$$

となり，これらを足し合わせたものがパラメータ数である．

$$\text{パラメータ数} = 1\text{ 層目} + 2\text{ 層目}〜n\text{ 層目} + \text{全結合層} + \text{バイアス項}$$

畳込み層を 4 層（n =4），フィルタを 256 枚（k =256）の場合は以下のようになる．

$$\text{パラメータ数} = 16,384 + 786,432 + 16,384 + 1,028 = 820,228$$

4.6　学習方法

本研究では，Matsuzaki[7] のプレイヤ（N-tuple ネットワークベース）による 2048 のプレイログ (平均得点約 46 万点) があるため，これを教師データとする教師あり学習を行う．扱ったデータは計 6 億盤面からなり，1 盤面には盤面（16 マス）のタイル情報と入力した方向がある．

本研究における CNN は各方向 i の確率 $P(i)$ を出力するため，教師データと同じ方向 i を選ぶ $P(i)$ を最大化するようにパラメータを調整する．最適化の目標として，確率 $P(i)$ を最大化することとクロスエントロピーを最小化することは同じであるため，クロスエントロピーを用いた次式が誤差関数 E である．$t(i)$ は教師データの正解ラベルであり，各方向に対して 0（不正解）か 1（正解）かの値を持つ．

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

最適化は上記の損失関数 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である「`tensorflow.train.AdamOptimizer`」によって行う．学習率はデフォルトの 0.001 とした [1].

バッチサイズは 1,000（盤面）で，0.1 億盤面の学習が完了する毎に直前の 1,000 盤面を用いて平均誤差と一致率を算出する．本来であればテストデータを用意すべきであるが，本研究において平均誤差と一致率よりも学習後の 1,000 ゲームによる平均得点による比較が重要であるため上記の方法とした．テストデータは間違えるとゲームオーバーに直結するような盤面が好ましいと考えており，テストデータの作成は今後の課題である．

4.7　学習後のプレイ方法

学習した CNN に対して「2048」をプレイさせる際に，現盤面のみを入力に与え，出力として得る 4 つの方向に対する割合が最も大きい 1 方向を入力方向とする方法を「1 盤面プレイ」と呼び，以下のプレイ方法を「8 盤面プレイ」と呼ぶこととする．

図 2.1 で示した通り，現在の盤面に対して 8 つの盤面を得ることができるため，8 つの盤面を 1 回ずつ学習した CNN に入力に与え，出力として 4 方向に対する割合を 8 セット得ることができる．それぞれの盤面で選ばれた 1 方向を 1 票として多数決制により入力方向を決める．例えば，上 4 票，右 3 票，下 1 票，左 0 票の場合は入力方向は「上」となり，上 3 票，右 3 票，下 2 票，左 0 票の場合は割合の合計値が大きい方を優先する．

表 4.1 はその例である．太文字はその盤面に対して最も高い割合であり，その横には現盤面に対応させた方向を表記している．この場合，下は 7 票で右は 1 票となり，多数決により下を選ぶこととなる．表の下段右から 2 番目を見るとその盤面（270° 回転）において，左に入力は論外であるが，右は最大の 37.2%で上は 31.7%，下は 31.2%と悩んだように受け取れる．一方で表の下段左から 1 番目を見ると最大は 67.3%で自信の差が大きくあるように感じられる．

元々は各方向の割合の合計値で比較して大きい方向を選ぶようにしたが，票数としては

4 : 3 : 1 : 0 であるのに，割合の合計値が 3 票の方向が大きいため選ばれた結果，悪手となりゲームーオーバーに繋がるプレイを確認したため，多数決制とした．しかしながら，同票の場合の選択が評価値の合計ではなく割合の合計であるため，上記の通り合計値が低いが入力すべき方向である場合も確認しているため，改善が必要である．

表 4.2 はプレイ方法による結果の違いを示している．同じ学習した CNN に対して入力に 1 盤面だけ与えるよりも 8 盤面与えた後に多数決とした方が遥かに優れていることがわかった．これは層数や Ch 数が異なる CNN 構成でも同様の結果が得られた．このため，6 章では 8 盤面プレイの結果のみを記載する．

表 4.1　対称性の 8 盤面に対する出力の例

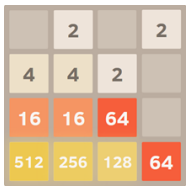
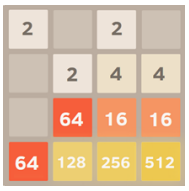
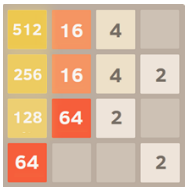
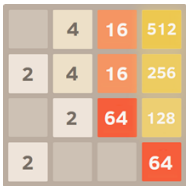
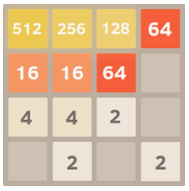
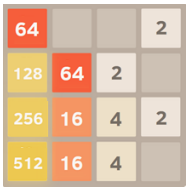
				
上	0.000	0.000	0.188	0.095
右	0.256	0.178	0.001	**0.644** (↓)
下	**0.660** (↓)	**0.560** (↓)	**0.444** (→)	0.252
左	0.084	0.261	0.367	0.009
				
上	**0.673** (↓)	**0.649** (↓)	0.317	0.248
右	0.083	0.147	**0.372** (↓)	0.001
下	0.000	0.001	0.312	0.196
左	0.244	0.202	0.000	**0.556** (↓)

表 4.2　5 層におけるプレイ方法による結果

プレイ方法	平均得点	最高得点	クリア率
1 盤面プレイ	24,105	182,296	35.8%
8 盤面プレイ	86,203	386,972	86.5%

第 5 章

提案 2：対称性を考慮した CNN

提案 1 における 8 盤面プレイを行う際の概略は図 5.1 であり，本節で述べる対称性を考慮した CNN の概略は図 5.2 である．

図 5.1 の赤破線枠が提案 1 の CNN を指しており，対称な 8 盤面の入力ひとつひとつが入力から出力まで同じ CNN を独立に通過する．従って，全結合層は 1 つの入力に対して 1 つあり，8 つの入力に対して各方向に対する組が 8 つ出力されることとなる．このため，8 つの各方向に対する組の出力を受けて CNN の判断を受けず，多数決というルールで 8 盤面プレイが提案 1 で行われた．

本節の提案である図 5.2 では対称な 8 盤面の入力ひとつひとつが同じ畳込み層を独立に通過するが，全結合層で対称な 8 盤面に対する特徴抽出の結果をまとめる．このため，1 つの各方向に対する組を出力するので，全結合層において多数決を担ってもらう構成となる．

実験では，1 盤面プレイと 8 盤面プレイを行うが提案 2 の構成上，1 盤面プレイが提案 1 でいう 8 盤面プレイに当たる．提案 2 における 8 盤面プレイとは図 5.2 における一番上の盤面を [「現盤面そのまま」,「現盤面を 0° 反転」,「現盤面を 90°」,「現盤面を 90° 反転」,「現盤面を 180°」,「現盤面を 180° 反転」,「現盤面を 270°」,「現盤面を 270° 反転」] と置き換えた後，それらの 0° 反転から 270° 反転までを対応させた 8 種類を入力に与えることで，8 通りの各方向に対する組を得て多数決を行う．

よって提案 2 における 1 盤面プレイと 8 盤面プレイの結果では，1 盤面プレイで既に対称な 8 盤面を CNN 内で考慮させるので同等の結果が得られることを期待する．

5.1　提案 1 との構成の違いについて

上記の通り，違いは全結合層のみである．全結合層への入力は，畳込み層の出力である 4×4 の k Ch を 8 盤面分であるため，4×4×k×8=256k を入力に与える．出力は同じである．

パラメータ数の算出は全結合層のみ変わり，全結合層のパラメータ数は次のようになる．

$$\text{全結合層} = (4 \times 4 \times Ch(k) \times 8) \times 4$$

5.1　提案 1 との構成の違いについて

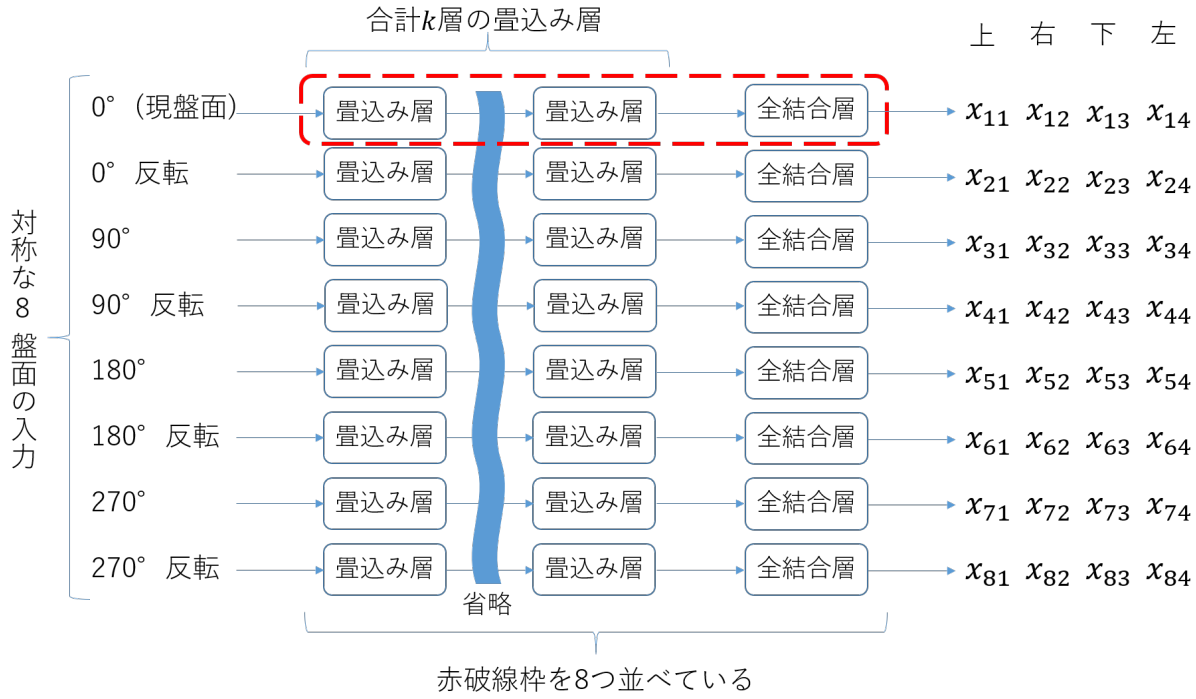


図 5.1　提案 1 の CNN で 8 盤面プレイを行う場合

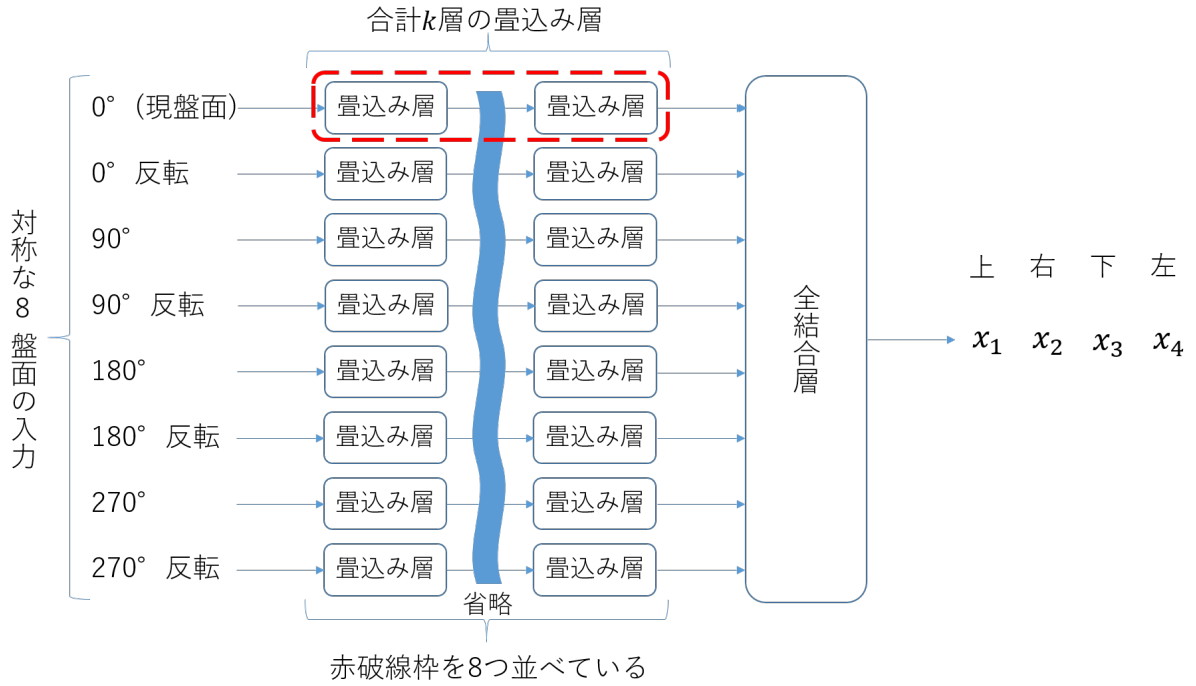


図 5.2　提案 2 の CNN の概要図

第 6 章

提案 1：対称性を考慮しない CNN を用いた実験

本章では，提案 1 の CNN 構成にて実験を行った．4.7 節で述べたようにプレイ方法は 8 盤面プレイのみを行った．また，本章の多くは著者の論文 [16] をまとめたものである．

6.1　畳込み層の層数による性能差

パラメータ数が約 82 万となる条件のもとで，畳込み層が 2 層から 9 層からなる 8 種類の CNN を構成し，教師データ 6 億盤面を与えて学習を行った．表 6.1 は作成した 7 種類の詳細である．

バッチサイズは 1,000（盤面）で，0.1 億盤面の学習をする毎にその時のパラメータの数値を保存した．表 6.2 は 5.9 億盤面から 6 億盤面にかけての学習において 6 億盤面に到達した際のパラメータで平均誤差と一致率を各層に分けて一覧にしたものである．平均誤差は 0 に近いほど良く一致率は 1 に近いほど良いため，2 層の平均誤差 0.698，一致率 0.692 が最も悪い結果となった．最も良いのが 5 層であるが，2 層と比べると 3 層から 9 層において大きな差ではない．

図 6.1 は 2 層と 5 層における学習盤面数に応じた平均誤差の推移を示している．3 層から 9 層はおおよそ 5 層と被る推移であった．

図 6.2 は 2 層，3 層，5 層における学習盤面数に応じた平均得点の推移を示している．これは 0.2 億盤面を学習する毎にその際のパラメータの数値で 1,000 ゲームを行いその平均得

点を集計したものである．2 層は学習盤面数が増加しても 2 万点前後を推移した．一方で 3 層と 5 層は学習盤面数が 0.2 億盤面から 1 億盤面で 3 万点から 5 万点まで一気に上昇し，それ以降は徐々に増加することがわかった．

表 6.3 は学習盤面数が 2 億，4 億，6 億に到達した際のパラメータで 1,000 ゲームを行ったときの平均得点，最高得点 (6 億時) ，クリア率 (6 億時) の結果を示したものである．6 億盤面の学習後の平均得点は 5 層の 86,203 点が最も良く，クリア率も 5 層の 86.5%が最も良かった．最高得点は 7 層の 401,912 点が最も良かった．また，傾向として 2 層から 5 層までは平均得点およびクリア率の上昇が見られたが，6 層以降は徐々に低下した．最高得点に関しては 1,000 ゲームの中で最高の 1 ゲームとなるため，たまたまうまく進んだ可能性がある点には注意が必要である．

表 6.4 は 6 億盤面の学習後のパラメータで，1,000 ゲーム行った際の最大タイルをまとめたものである．最大タイルとは，ゲームオーバー時にゲーム盤面にある一番大きな値のタイルのことである．例えば 2 層の 8192 が 1 というのは 1,000 ゲームの中で 8192 のタイルに到達したのが 1 ゲームということである．3 層から 9 層の CNN では 16384 のタイルに到達することができたが，32768 のタイルに到達することはできなかった．また，2048 のタイル以上の数値を足し合わせ，ゲーム数で割ったものがクリア率と一致する．

この調査において，畳込み層数が 2 層と 3 層の間には大きな差があることがわかった．この詳細に関しては，Matsuzaki らの内部解析を行った結果 [17] を参照のこと．

表 6.1　作成した 7 種類の CNN の層数，Ch 数，パラメータ数

畳込み層	Ch(k)	パラメータ数
2	436	817,068
3	312	818,688
4	256	819,200
5	222	818,074
6	200	825,600
7	182	818,272
8	168	811,776
9	158	819,072

表 6.2　5.9 億から 6 億の学習時の平均誤差と一致率

畳込み層	平均誤差	一致率
2	0.698	0.692
3	0.543	0.749
4	0.537	0.750
5	**0.528**	**0.758**
6	0.532	0.755
7	0.544	0.748
8	0.551	0.751
9	0.552	0.744

6.1　畳込み層の層数による性能差

図 6.1　2 層と 5 層における学習量に応じた平均誤差の推移

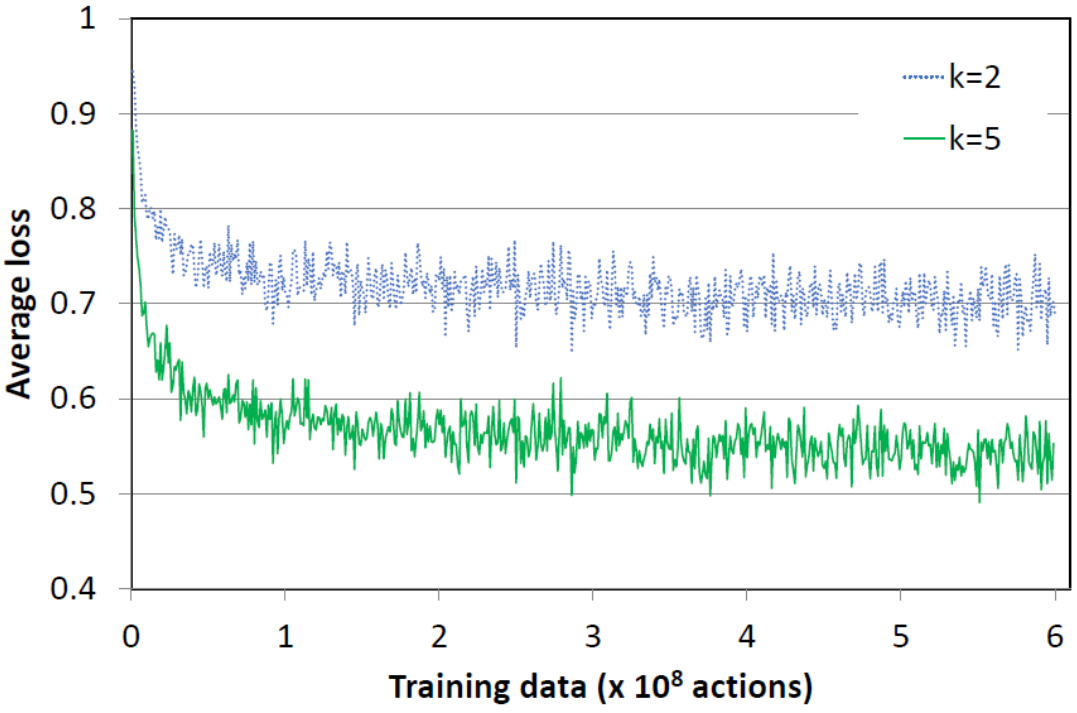


図 6.2　2 層，3 層，5 層における学習量に応じた平均得点の推移

6.1　畳込み層の層数による性能差

表 6.3　2 億，4 億，6 億盤面学習時の平均得点と最高得点およびクリア率

	平均得点			最高得点	クリア率
畳込み層	2 億	4 億	6 億	6 億	6 億
2	22,189	25,530	25,669	175,628	45.6%
3	61,037	64,212	69,840	332,868	79.4%
4	65,022	73,054	80,284	343,496	83.3%
5	**74,996**	**80,030**	**86,203**	385,560	**86.5%**
6	69,482	74,441	83,791	387,376	83.5%
7	67,874	77,448	79,812	**401,912**	83.1%
8	64,465	74,737	74,787	363,916	81.1%
9	66,732	73,484	68,129	358,736	75.9%

表 6.4　1,000 ゲームの最大タイル分布（異なる層数）

畳込み層	≤256	512	1024	2048	4096	8192	16384	32768
2	109	134	301	307	148	1	0	0
3	39	39	128	188	412	186	8	0
4	46	36	85	172	387	263	11	0
5	29	30	76	154	412	288	11	0
6	55	33	77	152	378	284	21	0
7	27	39	103	166	382	275	8	0
8	35	51	103	190	363	245	13	0
9	50	54	137	187	348	216	8	0

6.2　5 層におけるパラメータ数による性能差

前節において，畳込み層が 5 層の CNN が最も良い平均得点とクリア率を出した．このため，5 層においてパラメータ数を変化させた場合の調査を行った．

表 6.5 は作成した 5 種類の CNN の詳細である．Ch 数を 600 以上の場合は学習がうまく進まなかったことを確認した．

表 6.6 は学習後に 1,000 ゲーム行った結果の平均得点，最高得点，クリア率をまとめたものである．最も良かったのは Ch 数が 512 の構成で，平均得点は 118,434 点，最高得点は 386,812 点，クリア率は 89.5%となった．また，Ch 数が増えるごとに平均得点などが上昇する傾向があることがわかった．

表 6.7 は 1,000 ゲーム行った際の最大タイルの分布である．Ch 数の 222 と 512 を比較すると，クリア率は 3%の差であるが，16384 のタイルに到達できたゲーム数が約 7 倍となった．また，32768 のタイルに到達ができなかったことがわかった．

表 6.5　畳込み 5 層における Ch 数とパラメータ数

Ch(k)	パラメータ数
110	208,234
156	410,128
192	615,364
222	818,074
512	5,311,492

表 6.6　5 層で異なる Ch 数の CNN で 6 億盤面学習時の結果

Ch(k)	平均得点	最高得点	クリア率
110	44,056	255,620	66.0%
156	66,133	284,916	80.0%
192	77,426	289,968	82.8%
222	86,203	332,496	86.5%
512	118,434	386,812	89.5%

表 6.7　1,000 ゲームの最大タイル分布（同じ層数）

Ch(k)	≤256	512	1024	2048	4096	8192	16384	32768
110	50	89	201	282	316	60	2	0
156	54	54	92	198	406	192	4	0
192	44	44	84	154	409	261	4	0
222	29	30	76	154	412	288	11	0
512	33	25	47	81	349	395	70	0

第 7 章

提案 2：学習時に対称性を考慮した CNN での実験

本章では，提案 2 の CNN 構成にて実験を行った．

7.1　畳込み層数による性能差〜チャネル数を統一

表 7.1　Ch 数を 200 で統一し，畳込み層数の違いによる結果

	1 盤面プレイ			8 盤面プレイ		
畳込み層	平均得点	最高得点	クリア率	平均得点	最高得点	クリア率
3 層	46,405.6	305,712	55.4	49,863.5	320,772	56.8
4 層	49,334.8	332,480	56.1	55,521.4	274,224	61.7
5 層	80,354.6	333,848	72.7	86,061.7	366,504	71.9
6 層	107,757.1	428,752	86.9	108,484.7	508,328	85.2

畳込み層を 3 層，4 層，5 層，6 層の 4 種類で各 Ch 数を 200 で統一という条件で比較を行う．学習には同じ教師データを 6 億盤面与え，それぞれに対して学習後に 1 盤面プレイと 8 盤面プレイを 1,000 ゲーム行った．

表 7.1 はその結果である．1 盤面プレイと 8 盤面プレイの平均得点の差はどれも 1 万点未満であった．1 盤面プレイについて，3 層，4 層の平均得点は 4 万点台であるのに対して 5 層では 8 万点台と 2 倍近くの差があった．また，クリア率を見ても 3 層，4 層では 2048 の

タイルに到達できない割合が 5 割近いことがわかる．6 層は平均得点が最も高く，最高得点とクリア率に関しても優れた．このことから，Ch 数を統一した場合は畳込み層数が多い方が性能が良いことがわかった．

表 7.2　異なる畳込み層数と同じ Ch 数でのパラメータ数一覧

畳込み層	Ch(k)	パラメータ数
3 層	200	444,804
4 層	200	608,004
5 層	200	771,204
6 層	200	934,404

この結果の要因の一つとして総パラメータ数が異なることが挙げられる．

表 7.2 は本節の 4 種類における総パラメータの一覧である．総パラメータ数が多いほど平均得点，最高得点，クリア率が高くなる傾向があった．

また，1 層増えるごとにパラメータ数が 163,200 増えており，これは畳込みフィルタサイズが 2×2 で畳込み層における入力数と出力数が Ch 数と同数の 200 となるため，2×2×200×200=160,000 であり，これに出力に加えるバイアス項の 2×2×200=3,200 を足したものとなる．

7.2　畳込み層数による性能差～パラメータ数を統一

表 7.3　異なる畳込み層数と異なる Ch 数でのパラメータ数一覧

畳込み層	Ch(k)	総パラメータ数
3 層	415	1,636,764
4 層	343	1,631,312
5 層	300	1,636,804
6 層	275	1,639,444

表 7.4　異なる畳込み層数で，同等のパラメータ数による結果

		1 盤面プレイ			8 盤面プレイ		
畳込み層	Ch(k)	平均得点	最高得点	クリア率	平均得点	最高得点	クリア率
3 層	415	65,858	341,416	74.4%	68,932	352,956	75.5%
4 層	343	65,275	358,308	60.7%	71,686	338,320	66.1%
5 層	300	85,548	386,744	72.2%	93,964	405,600	80.5%
6 層	275	測定不可					

前節では，パラメータ数が多いほど平均得点，最高得点，クリア率が優れることがわかった．そこで，畳込み層を 3 層から 6 層で総パラメータ数を約 163 万で統一してみる．

表 7.3 はその一覧である．学習には同じ教師データを 6 億盤面与え，それぞれに対して学習後に 1 盤面プレイと 8 盤面プレイを 1,000 ゲーム行った．

表 7.4 はその結果である．

6 層に関しては，学習開始直後である 10 万盤面を学習した段階で一致率が 25%前後 (4 方向から正解 1 方向をランダムに選ぶ確率と同じ) となり，以降学習を進めても改善せず中断し，測定不可とした．また，それ以上にパラメータ数を増加した場合においても同様の結果が得られたため，今回の CNN 構成における 6 層でのパラメータ数の増加はできないことがわかった．

7.3　畳込み 5 層における異なる Ch 数による性能差

表 7.5　同じ畳込み層数と異なる Ch 数のパラメータ数一覧

畳込み層	Ch(k)	パラメータ数
5 層	100	225,604
5 層	200	771,204
5 層	300	1,636,804
5 層	400	2,822,404
5 層	500	4,328,004

表 7.6　畳込み層数 5 層で統一し，Ch 数の違いによる結果

		1 盤面プレイ			8 盤面プレイ		
畳込み層	ch(k)	平均得点	最高得点	クリア率	平均得点	最高得点	クリア率
5 層	100	43,552	331,748	54.7%	45,710	330,756	55.5%
5 層	200	80,355	333,848	72.7%	86,062	366,504	71.9%
5 層	300	85,548	386,744	72.2%	93,964	405,600	80.5%
5 層	400	119,637	533,624	80.7%	109,493	509,284	77.3%
5 層	500	測定不可					

前節で，同等のパラメータ数においては 5 層が最も優れていることがわかった．そこで，同じ層数においてチャネル数を変化させた場合の違いを確認する．本節までに Ch 数が 200 と 300 は結果がわかっているため，新たに 100，400，500 の学習を行った．

表 7.5 は Ch 数とパラメータ数の一覧である．学習には同じ教師データを 6 億盤面与え，それぞれに対して学習後に 1 盤面プレイと 8 盤面プレイを 1,000 ゲーム行った．

表 7.6 はその結果である．畳込み層 5 層の Ch 数 400 の 1 盤面プレイの平均得点は 119,637 点で，最高得点は 533,624 点と本研究において最も高い数値となった．しかしながら，クリア率は最も高い数値とはならなかった．

7.4　教師データの学習順と乱数 seed 値による影響

まず，本研究における教師データは 60 ファイルの計 6 億盤面あり，各ファイルはランダムに並び替えられた連続しない 0.1 億盤面を持つ．これまでの実験では学習時にはファイル 0 からファイル 59 までを順に与えた．この学習時に与えるファイル順によって，学習が偶然うまく進み結果が良くなっている可能性やその逆の可能性がある．

次に，本研究における学習時のプログラムの乱数は TensorFlow の変数の初期値設定がある．これまでは「20170412」という固定値を「`tensorflow.set_random_seed(20170412)`」として設定していた．この TensorFlow の変数の初期値によって，学習が偶然うまく進み結果が良くなっている可能性やその逆の可能性がある．

そこで，同じ構成の CNN を用いて，教師データの学習順と seed 値を変化させた場合の比較を行う．ここで用いる CNN は，畳込み 5 層 (各 Ch 数 400) である．

ただし，学習率をこれまで通り 0.001 とすると学習がうまく進まず，学習率は 0.0001 とした．この学習率の設定は「`tensorflow.train.AdamOptimizer(learing_rate=0.0001)`」により行った．

また，学習順は昇順，降順，ランダム 5 種の 7 通りとする．ランダム 5 種は「`numpy.random.seed()`」の seed 値を 1 から 5 まで (それぞれ np1 から np5 とする) であり，60 ファイルが被りなしで各 seed 値に対応した順とする．なお設定には「`numpy.random.choice(index, 60, replace=False)`」(index は numpy の配列で 0 から 59 の添字と中身が対応させたもの) とした．

学習後は 1 盤面プレイと 8 盤面プレイを 1,000 ゲーム行った．

表 7.7 は教師データの学習順と乱数 seed 値が異なる 7 通りを行った結果である．1 盤面プレイおよび 8 盤面プレイの平均得点は ±5000 程であった．つまり教師データの学習順による性能に大きな差は見られず，また TensorFlow の変数の初期値における乱数 seed 値による性能大きな差は見られなかった．このことから教師データの学習順と乱数 seed 値による影響はないといえる．

表 7.7　学習順と seed 値の変化による結果

		1 盤面プレイ			8 盤面プレイ		
学習順	seed 値	平均得点	最高得点	クリア率	平均得点	最高得点	クリア率
np1	1	93,695	479,136	79.8%	93,295	528,944	77.9%
np2	2	99,768	387,336	83.5%	98,036	434,340	82.5%
np3	3	99,427	630,188	81.7%	97,323	509,116	80.7%
np4	4	95,315	404,520	81.4%	100,517	584,708	82.3%
np5	5	96,185	404,964	80.%7	98,204	483,176	81.2%
昇順	20170412	99,070	403,844	81.5%	97,091	499,492	80.9%
降順	20170412	103,050	543,560	84.6%	99,935	547,712	83.4%
np1～np5 平均		96,878	461,228	81.4%	97,475	508,056	80.9%
全体平均		98,073	464,792	81.9%	97,772	512,498	81.3%

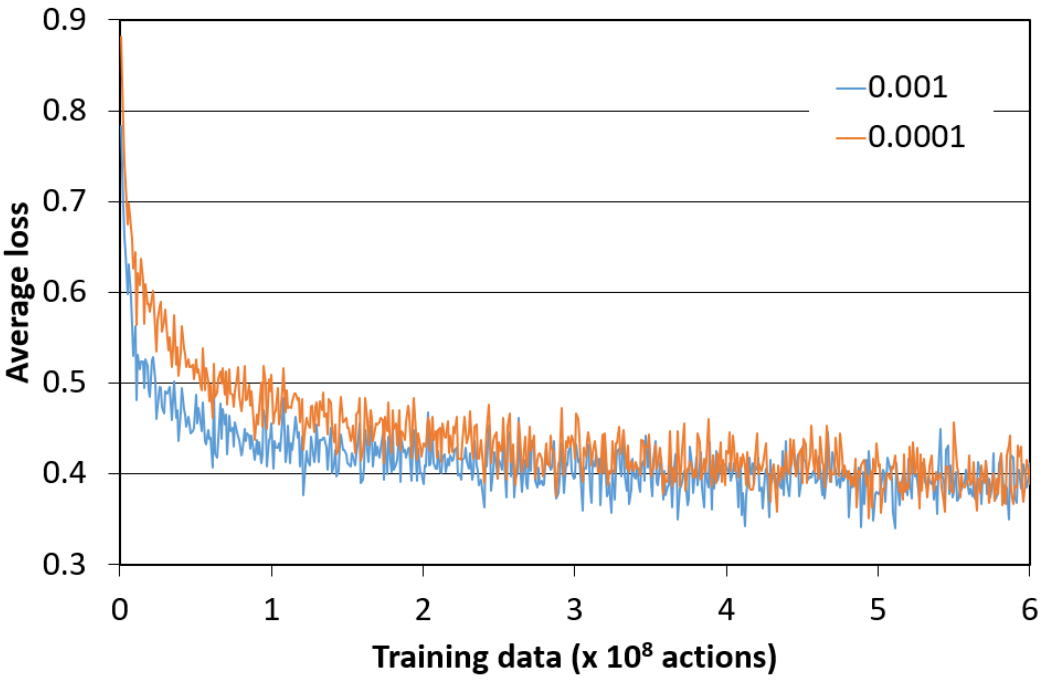


図 7.1　2 種の learning_rate における平均誤差の推移

7.4 教師データの学習順と乱数 seed 値による影響

次に，本節における実験のため学習率を 0.001 から 0.0001 に下げた影響について確認する．学習率 0.001 の 1 盤面プレイの平均得点は表 7.6 の通り 119,637 点であり，学習率 0.0001 の 1 盤面プレイの平均得点は表 7.7 の昇順の行の通り 99,070 点であり，2 万点程の差が生じた．

図 7.1 は 2 種の学習率における平均誤差の推移である．横軸の 0 盤面 (学習開始) から 1 億盤面の推移に着目すると平均得点が高かった学習率 0.001 の方が低い (早く学習が進んだ) ことがわかる．これは学習率が高いためより学習がうまく進んだということである．右端の 6 億盤面だけ見るとほぼ同じ値になっているが，学習率 0.001 は 3 から 6 にかけて横這いであるが，学習率 0.0001 は徐々に減少していて，学習量を 3 億程追加すると減少せず横這いに落ち着きそうである．このことから 6 億盤面を学習した時点では，平均得点に 2 万点程の差があるが，追加学習を行えば学習が十分となり同等の結果になると予想する．

学習率を 0.001 から変化させた実験は本節が初めてであり，これまで行ってきた実験のうち，パラメータ数を増やした際に学習がうまく進まないことが多々あったが，学習率を下げることで改善する可能性があると考える．パラメータ数を増やした CNN で学習がうまく進めば平均得点は向上する傾向にあるため，学習率の調整を行うことは今後の調査課題である．

第 8 章

提案 1 と提案 2 の比較

6 章では提案 1 の CNN 構成を用いて実験を行い，7 章では提案 2 の CNN 構成を用いて実験を行った．本章では，それぞれで最も平均得点が高かった CNN 構成の結果について比較し，考察する．

8.1 結果比較

表 8.1 は 6 章（提案 1）と 7 章（提案 2）の最良モデルの構成と結果をまとめたものである．6 章では提案 1 の CNN 構成において，畳込み層を 5 層，Ch 数を 512 としたものが最良モデルであり 8 盤面プレイによって行った結果であり，7 章では提案 2 の CNN 構成において，畳込み層を 5 層，Ch 数を 400 としたものが最良もであるであり 1 盤面プレイによって行った結果である．平均得点に関しては約 1,000 点程，提案 2 のものが高かった．これは再度集計した場合において入れ替わる範囲の差である．最高得点に関しては約 147,000 点程，提案 2 ものが高かった．クリア率に関しては 8.8%，提案 2 のものが低かった．

表 8.2 は 1,000 ゲーム行わせた際の最大タイル分布をまとめたものである．提案 2 のものでは 32768 のタイルに 1 度到達できたことがわかった．16384 のタイルに到達した後にゲームオーバーになったゲーム数は提案 1 と提案 2 のもので 50 ゲームも差があった．一方で，512 のタイルに到達できなかった（256 以下の）ゲーム数は提案 1 と提案 2 で 104 ゲームもの差があった．このことから，平均得点では大きな差が見られなかったが，提案 1 のものは序盤（512 のタイルができるまで）でのミスが少なく安定したプレイができており，提案 2 のものは序盤以降でのゲームオーバーを避けるプレイができていることがわかる．

8.1　結果比較

図 8.1 は学習段階毎の平均誤差の推移を示したものである．提案 1 は 0.5 前後で，提案 2 は 0.4 前後の推移となった．

表 8.1　提案 1 と提案 2 の各最良モデルの構成詳細と結果

	畳込み層	Ch(k)	パラメータ数	平均得点	最高得点	クリア率
提案 1	5 層	512	5,311,492	118,434	386,812	89.5%
提案 2	5 層	400	2,822,404	119,637	533,624	80.7%

表 8.2　提案 1 と提案 2 の各最良モデルの 1,000 ゲームの最大タイル分布

	≤256	512	1024	2048	4096	8192	16384	32768
提案 1	33	25	47	81	349	395	70	0
提案 2	137	39	17	53	275	358	120	1

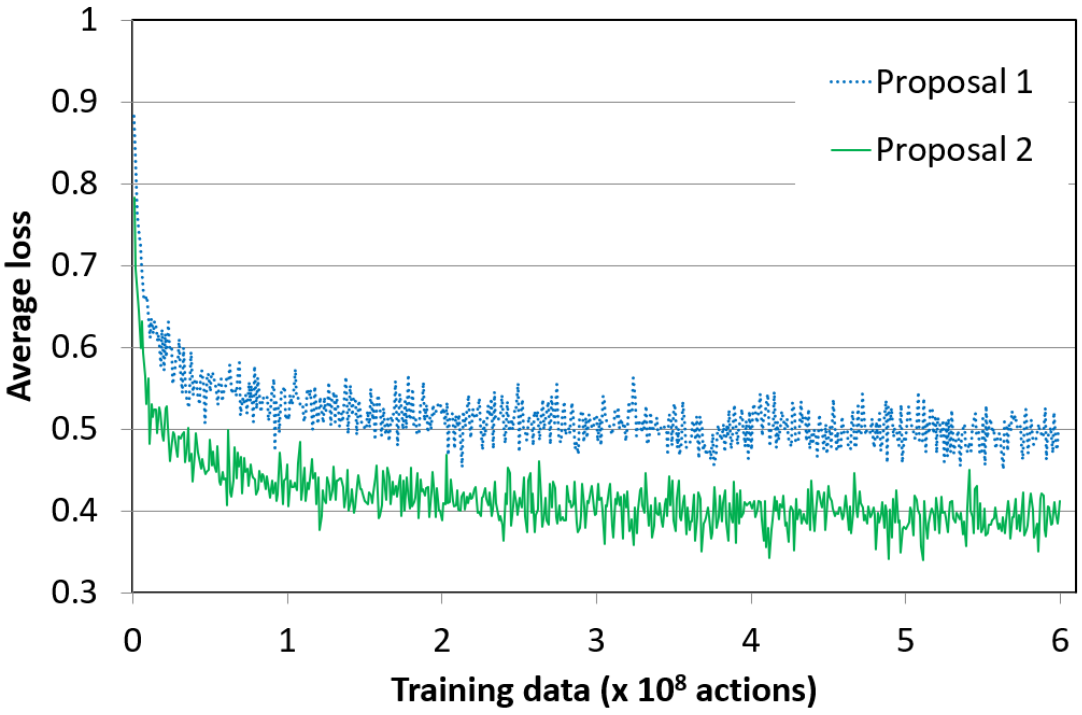


図 8.1　提案 1 と提案 2 の各最良モデルでの平均誤差の推移

8.2 考察

ゲーム「2048」における性質について考える．表 8.3 は特定のタイルがゲーム盤面上に併合によって初めてできる際のおおよその合計得点と手数をまとめたものである．ゲームのクリアとなる 2048 のタイルに到達するまでにはおよそ 1,100 手かかる．一方で，クリア以降の 32768 のタイルに到達するまでには 15 倍の手数が必要になる．このことを念頭において，前節の結果を振り返る．提案 2 の最良モデルでは 512 のタイルに到達できずにゲームオーバーとなったのが 137 ゲームあった．つまり，約 300 手未満でゲームオーバーとなったこととなる．教師データは 6 億盤面（6 億手数）分あるが，これは約 33,000 ゲームからなり，300 手以下の教師データは全体の 1.65%しかない計算となる．このことから，提案 1 の CNN 構成では 1.65%の序盤のデータをうまく取り込めていたが，提案 2 の CNN 構成ではうまく取り込めなかったと考えられる．よって，序盤のデータを増やすことで提案 2 の CNN 構成の平均得点とクリア率が向上する可能性がある．一方で，提案 2 の CNN 構成では残りの 98.45%の教師データに関しては提案 1 の CNN 構成よりうまく取り込むことができたと考えられる．前節の図 8.1 はこのことを良く示しており，平均誤差の差の要因と推測する．

また，N-tuple ネットワークのコンピュータプレイヤではマルチステージ化というアイデアがある．これは A のタイルができるまではプレイヤ A を用い，B のタイルができるとプレイヤ B にバトンタッチするというものである．本研究において序盤は提案 1 の構成が優れ，序盤以降は提案 2 の構成が優れているため，512 のタイル前後をしきいとしてマルチステージを行えば，序盤でゲームオーバーになるゲームの得点を大幅に伸ばせる可能性があり，平均得点の向上が期待できると考えられる．

表 8.3　あるタイルが初めてできる際のおおよその合計得点と手数

タイル	合計得点	手数
256	2,000	170
512	4,500	300
1024	10,000	600
2048	21,000	1,100
4096	46,000	2,000
8192	100,000	4,000
16384	210,000	7,600
32768	460,000	15,400

第 9 章

まとめ

本論文では，大きく分けて 2 種類の CNN 構成を提案した．提案 1 の CNN 構成では，先行研究のゼロパディングなしにより構成できる畳込み層の層数に上限があることに着目し，ゼロパディングありにすることによって畳込み層の層数を構成できる上限をなくし，層数の違いによる性能の差を示した．この際に，現盤面だけをプレイ時に渡す 1 盤面プレイより，現盤面を回転・反転により得られる対称な 8 盤面をプレイ時に渡す 8 盤面プレイの方が断然性能が良いことを示した．これにより，提案 2 の CNN 構成では学習後のプレイ時に対称な 8 盤面を渡すだけでなく，学習時から対称性を取り込める構成とした．提案 2 の構成の学習は成功し，平均得点は 119,637 点，最高得点は 533,624 点，クリア率は 80.7%であった．また 32768 のタイルに到達することができていた．一方で提案 1 の構成では，平均得点は 118,434 点，最高得点は 386,812 点，クリア率は 89.5%で，平均得点に差はないがクリア率が 8.8%高い結果となっていた．提案 2 の構成では序盤に弱く，序盤以降は強いことがわかり，提案 1 はその逆であることを示した．提案 2 の序盤に弱いという点は序盤の学習データを増やすことで改善できると考えられるため，提案 2 の対称性を考慮した CNN の構成は有用な手法であったと結論付ける．

謝辞

本研究を行うにあたり，指導教員である高知工科大学情報学群の松崎公紀教授には研究室配属から 4 年間に渡り，研究活動や論文執筆，プレゼンテーションなどにおいて多くのご指導をいただきました．熱心なご指導のおかげで香港理工大学や台湾国立交通大学へ伺う機会が得られ，そこでの学生交流は忘れられないものとなりました．心より感謝申し上げます．また，本論文の副査を引き受けていただいた吉田真一准教授，竹内聖悟講師に心からお礼申し上げます．最後に，本研究を遂行する上で，その実験の多くを高知工科大学の IACP クラスタ計算機を使用させていただきました．本当にありがとうございました．

参考文献

[1] 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).

[2] Silver, D. et al.,: A general reinforcement learning algorithm that masters chess, shogi, and Go through self–play, Science, vol.362, No.6419, pp.1140–1144 (2018).

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

[4] Guei, H., Wei, T., Huang, J.-B. and Wu, I.-C.: An Early Attempt at Applying Deep Reinforcement Learning to the Game 2048.

[5] 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).

[6] 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).

[7] Matsuzaki, K.: Developing 2048 Player with Backward Temporal Coherence Learning and Restart, Proceedings of Fifteenth International Conference on Advances in Computer Games (ACG2017), pp.176–187 (2017).

[8] 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).

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

[10] 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).

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

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

[13] 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).

[14] 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).

[15] 内田 祐介, 山下 隆義, 物体認識のための畳み込みニューラルネットワークの研究動向, 電子情報通信学会論文誌 D, Vol.J102-D, No.3, pp.203–225 (2019).

[16] Kondo, N. and Matsuzaki, K.: Playing Game 2048 with Deep Convolutional Neural Networks Trained by Supervised Learning, Journal of Information Procssing, Vol. 27, pp.340–347 (2019).

[17] Matsuzaki, K. and Teramura, M.: Interpreting Neural-Network Players for Game 2048, Proc. 2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI 2018), pp.136–141 (2018).