令和 2 年度

修士学位論文

ガイスターにおいてプレイ履歴から駒色推定を行うアルゴリズムの研究

Study of a Play History Based Hidden Information Estimation Algorithm in Geister

1235065　　寺村　舞童華

指導教員　　松崎　公紀

2021 年 3 月 1 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

ガイスターにおいてプレイ履歴から
駒色推定を行うアルゴリズムの研究

寺村　舞童華

ガイスターとは，アレックス・ランドルフが作成し 1982 年に発売された 2 人用ボードゲームである [1]．ガイスターは，相手の駒色が分からない状態でゲームが行われる二人零和確定不完全情報ゲームである．一般的に不完全情報ゲームにおいて隠された情報の推定を行うことは重要とされている．ガイスターでの色推定を行う研究として，モンテカルロ法を用いた研究 [3][4] があり，色推定の有効性について研究されている [5]．また，機械学習を用いた研究 [6] もある．

本研究では，畳み込みニューラルネットワーク (以下 CNN と呼ぶ) を用いて相手の駒色を推定する方法を提案し，その有効性を実験により評価する．CNN の学習と評価には，モンテカルロ法ベースのプレイヤのプレイ履歴を用いた．

キーワード　ガイスター, 機械学習, 畳み込みニューラルネットワーク

Abstract

Study of a Play History Based Hidden Information Estimation Algorithm in Geister

TERAMURA Madoka

Geister is a two-player board game created by Alex Randolph and released in 1982[1]. Geister is a two-person zero-sum confirmed incomplete information game in which the game is played without knowing the opponent's piece color.In general, it is important to estimate hidden information in imperfec t information games. As a study for color estimation in Geister, there is a study using the Monte Carlo method [3][4], and the effectiveness of color estimation is being studied [5]. There is also a study using machine learning [6]. In this study, we propose a method to estimate the opponent's piece color using a convolutional neural network (hereinafter referred to as CNN), and evaluate its effectiveness experimentally. The play history of Monte Carlo-based players was used for learning and evaluation of CNNs.

key words Geister, MachineLearning, Convolutional Neural Network

目次

図目次

表目次

第 1 章

はじめに

2017 年からガイスター AI の大会が開催されており [2]，ガイスターの研究は盛んに行われてきた．ガイスターの色推定を行う研究としてモンテカルロ法を用いたプレイヤ (以下モンテカルロプレイヤと呼ぶ) の研究 [3][4] があり，色推定の有効性についての研究 [5] も行われている．ガイスターにおいて隠された情報を推定することでゲームを有利に進めることができ勝率も上がると考えられる．また，機械学習を用いた研究 [6] もあり，色推定を行うアルゴリズムとして機械学習を行う手法を用いて有効性を示したいと考えた．そこで本研究では，畳み込みニューラルネットワーク (以下 CNN と呼ぶ) を用いて相手の駒色推定を行う方法を提案する．学習と評価にはモンテカルロプレイヤのプレイ履歴を用いた．

本論文は次のとおり構成されている．第 2 章ではガイスターについてゲームのルールと，ガイスターにおける AI プレイヤーについて記述する．第 3 章では，本研究で用いた技術の CNN について説明し，本研究における学習のための予備実験について記述する．第 4 章から第 6 章で予備実験から CNN の構成を変え実験を行った結果と考察を記述する．第 7 章で関連研究について記述し，第 8 章で本論文をまとめる．

第 2 章

ガイスター

2.1　ガイスターのルール

ガイスターは，縦横 6×6 の盤面でゲームが行われ四隅に脱出口と呼ばれるものがある．一人 8 個の駒を持っていて，それぞれ青駒が 4 つ，赤駒が 4 つずつ持っている．上下左右のどちらか一方に駒を一つ動かすことができ，自分の脱出口を目指して駒を動かしていく．盤面の四隅にある脱出口のうち，相手の陣地側にあるものが自分の脱出口である．ゲーム開始の初期配置を図 2.1 に示す．図 2.1 において，囲まれている 2×4 の範囲で自分の駒をランダムに配置する．プレイヤは相手の駒色がわからない状態でゲームを行い，相手の駒を取ることで相手の駒色を知ることができる．

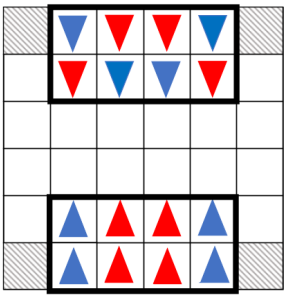

図 2.1　ガイスターの初期盤面

またプレイヤは，以下の条件一つを満たすことで勝利できる．図 2.2 はそれぞれ勝利条件を満たした場合の盤面を表している．

(1). 相手の青駒をすべて取る

(2). 相手に赤駒をすべて取らせる

(3). 相手の陣地にある自分の青駒を脱出口から脱出させる

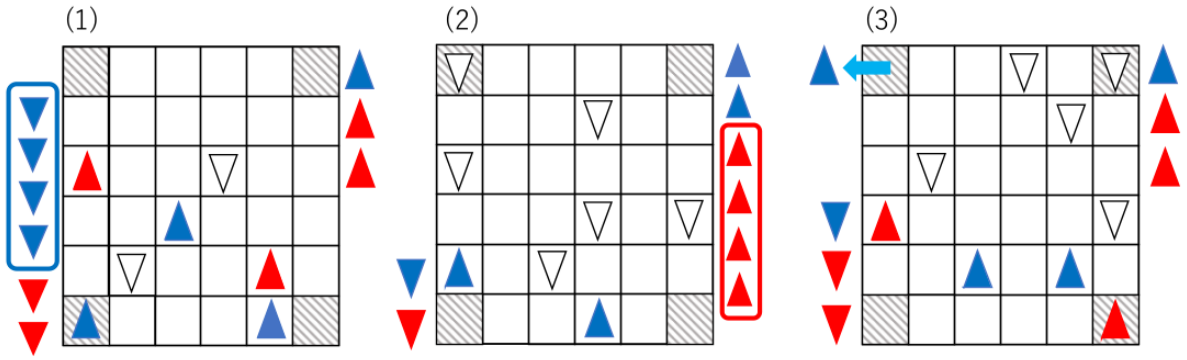

図 2.2　ガイスターの勝利時の盤面

2.2　ガイスタープレイヤについて

これまでガイスター AI の研究は盛んに行われており，いくつかのガイスタープレイヤが作成されている．本研究のネットワークの学習に用いたプレイ履歴の元になっているモンテカルロプレイヤ [3] と，機械学習を用いたプレイヤ [6] などがある．ここでは，モンテカルロプレイヤについて説明する．

[3] のプレイヤは，Using Past Playout(以下 UPP と呼ぶ) と呼ばれる相手の隠された情報の推定を行うアルゴリズムを用いて駒色推定を行っているプレイヤである．UPP は，モンテカルロ法のシミュレーションの内相手が着手した駒色を仮定し合法手の中で勝率の比較を行う．相手の勝率が最も高くなった場合の駒色が実際の駒色と等しいとして，相手の着手から駒色の推定を行う．UPP を用いたモンテカルロプレイヤと単純なモンテカルロプレイヤとの対戦結果は，UPP を用いたモンテカルロプレイヤが 55%という結果になっている．

第 3 章

畳み込みニューラルネットワークについて

3.1　CNN について

CNN とは，機械学習の一つで画像認識や動画認識の分野で使われ優れた性能を持つネットワークであり，広い分野に用いられている．このネットワークでは，主な機能である畳み込み層とプーリング層と呼ばれる層を重ねたもので構成されている．

畳み込み層では，入力データに対して畳み込みフィルタを用いて畳み込み処理を行い，ある画像からの特徴を抽出することが可能である．プーリング層では，畳み込み層から得られた特徴が強調されるように処理を行う．画像認識ではカテゴリ分類を行うことが重要であるため物体の写っている位置についての情報をプーリング層で落とす役割を持っている．CNN を用いることで，画像の中の物体が何かカテゴリとして分類することが可能である．

3.2　予備実験

3.2.1　CNN の構成

図 3.1，図 3.2，図 3.3 は，予備実験で用いる各 CNN の構成を示している．各図においてフィルタの数を 128 としており，入力データに対して畳み込みを行われた後の出力は 6×6×128 となる．それぞれ図 3.1 のモデルをモデル 1，図 3.2 のモデルをモデル 2，図 3.3 のモデルをモデル 3 と呼ぶ．表 3.1 はそれぞれのモデルの違いについての対応を表してい

る．次節からそれぞれ入力層，畳み込み層，全結合層，出力層，パラメータについて細かく説明する．

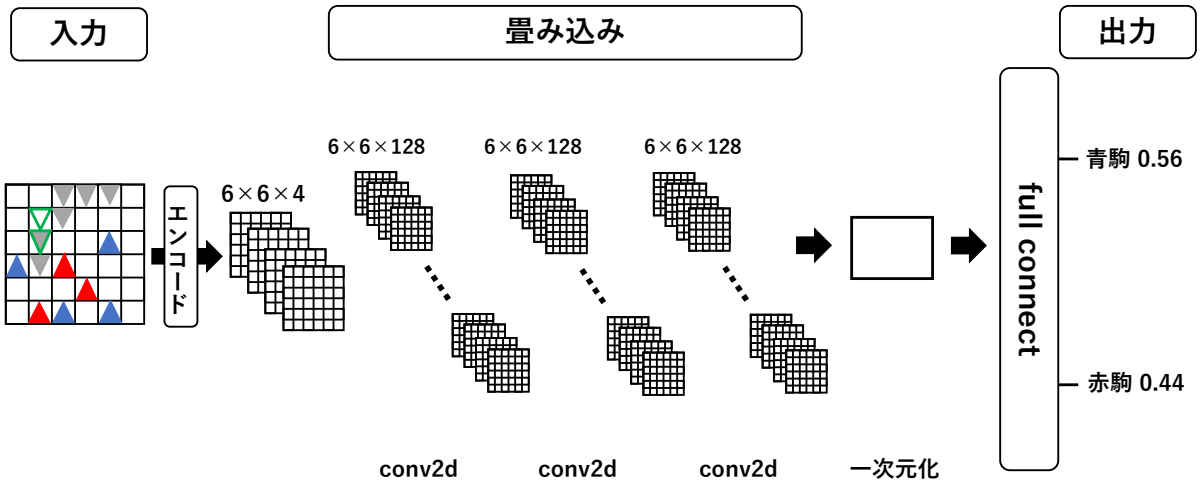

図 3.1　モデル 1 の CNN の構成

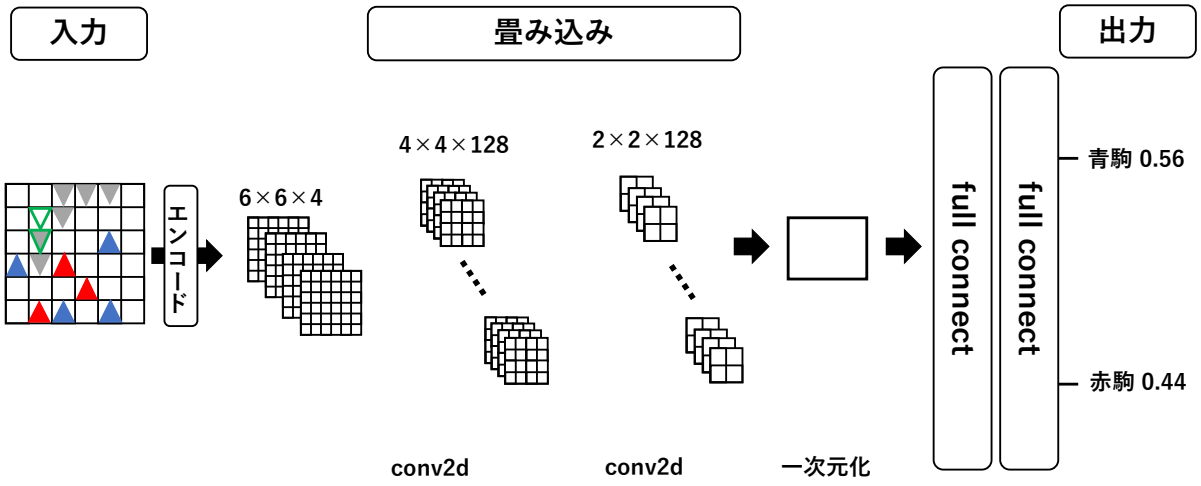

図 3.2　モデル 2 の CNN の構成

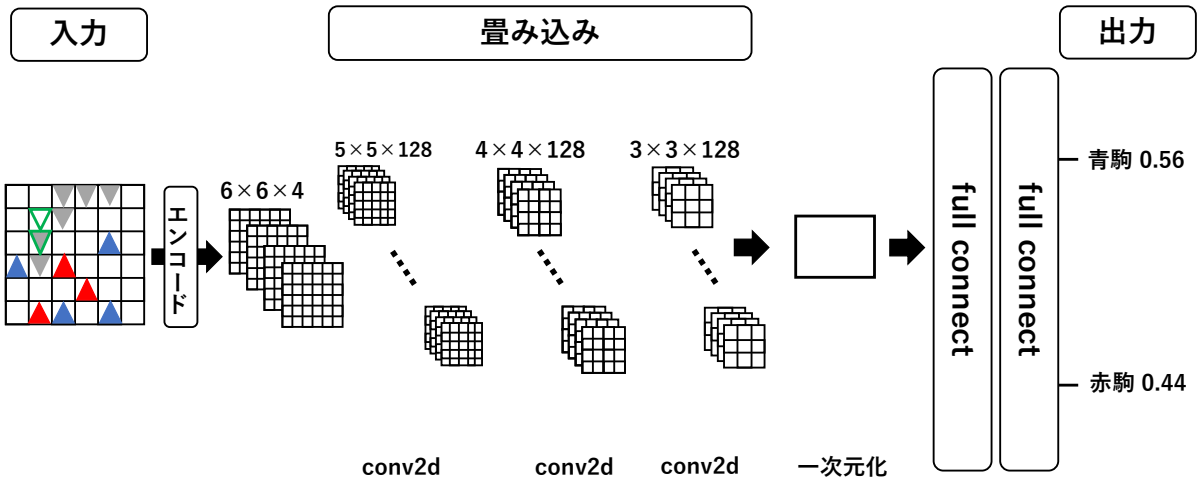

図 3.3　モデル 3 の CNN の構成

表 3.1　予備実験で用いたモデルの違い

モデル	フィルタサイズ	畳み込み層	全結合層	パディング
モデル 1	3×3	3	1	あり
モデル 2	3×3	2	2	なし
モデル 3	2×2	3	2	なし

3.2.2　入力層

各モデルの入力層に入力されるデータに違いはなく，6×6 の Ch 数 4 のものが入力される．これは，与えられた盤面の縦 × 横 (6×6) と，相手駒の位置，自分の青駒の位置，自分の赤駒の位置，推定する相手駒の位置の移動元と移動先のデータをそれぞれについてエンコードしたものである．エンコードは，相手駒の位置，自分の青駒の位置，自分の赤駒の位置は同様で盤面にあれば 1，なければ 0 とする．想定する相手駒の位置では，移動元を-1，移動先を 1 としてエンコードされる．

3.2.3　畳み込み層

畳み込み層では，モデル 1 とモデル 2 はフィルタサイズ 3×3 のもの，モデル 3 ではフィルタサイズ 2×2 のものを k 枚用いる．フィルタの数は，各モデルの総パラメータ数が統一されるように調整を行い決定する．モデル 1 はゼロパディングあり，ストライド 1 で畳み込みを 3 回行う，モデル 2 はパディングなし，ストライド 1 で畳み込みを 2 回行う．モデル 3 はパディングなし，ストライド 1 で畳み込みを 3 回行う．1 層目の畳み込み層の結果は，畳み込み結果に対して，バイアス項を加え，活性化関数の ReLU を適用したものである．

3.2.4　全結合層

畳み込み層からの出力を一次元化したものを入力に与える．それぞれ青駒と赤駒で推定した結果の出力とするため 2 とし，この出力結果にバイアス項を加えたものが全結合層の出力となる．

3.2.5　出力層

全結合層からの出力の 2 つの値を softmax 関数により青駒と赤駒について推定した値を確率に変換する．

3.2.6　パラメータ数

提案した CNN のパラメータ数は以下のように算出している．モデル 3 のフィルタサイズ 2×2 でゼロパディングなしの場合とする．

畳込み層 1 層目はフィルタサイズ 2×2 を Ch 数 4 に対して k 枚のフィルタを掛けるため

$$1\text{層目} = (2 \times 2 \times 4) \times Ch(k)$$

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 2 層あり，1 つ目は

$$1\text{ つ目の全結合層} = (3 \times 3 \times Ch(k)) \times Ch(k)$$

2 つ目の全結合層では，出力は 2 つとするため

$$2\text{ つ目の全結合層} = Ch(k) \times 2$$

バイアス項は 1 つ目の全結合層で Ch(k)，2 つ目で 2，畳込み層では 1Ch につき 1 つを n 層のため

$$\text{バイアス項} = Ch(k) + 2 + Ch(k) \times n$$

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．

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

畳込み層を 3 層（n =3），フィルタを 128 枚（k =128）の場合は以下のようになる．

$$\text{パラメータ数} = 2,048 + 131,072 + 147,456 + 256 + 514 = 281,346$$

3.2.7　実験

プレイ履歴から 30 万局面を 1000 バッチごとに学習させ性能の差を比較した．学習モデルの総パラメータ数は 30 万程度になるように調整を行っている．

学習時，出力の確率 $P(i)$ を最大化させるためにクロスエントロピーを最小化することを目指し，クロスエントロピーを用いる次式を誤差関数 E とした．$t(i)$ 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．

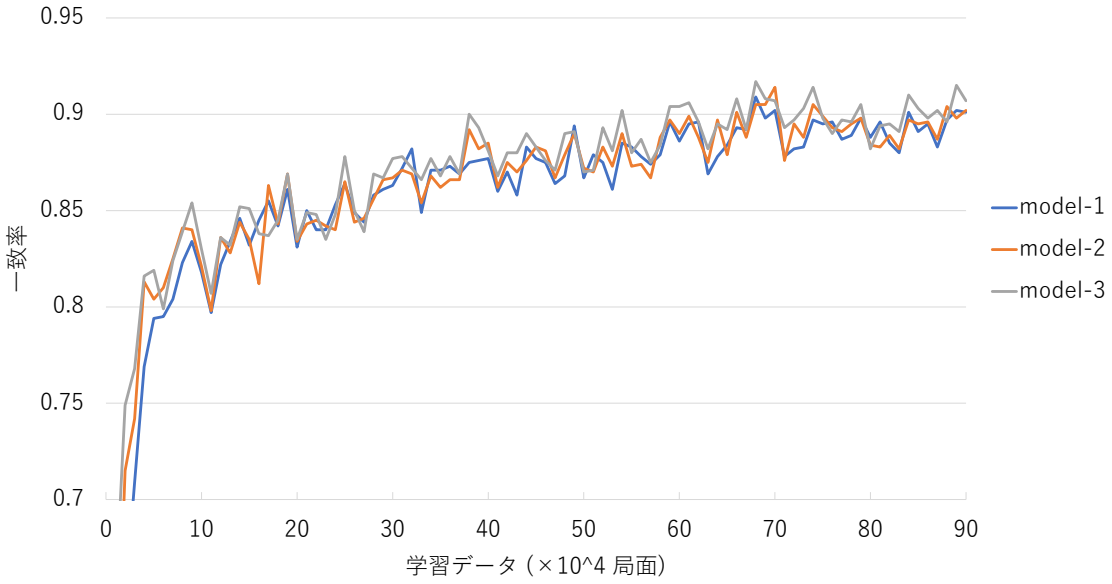

図 3.4　予備実験の結果

3.2.8　結果

図 3.4 は予備実験の結果を表している．グラフの凡例について，「model-1」はフィルタサイズが 3×3 でパディングありのモデルを表し，「model-2」はフィルタサイズが 3×3 でパディングなしのモデルを表し，「model-3」はフィルタサイズが 2×2 のモデルを表している．実験結果は，図 3.4 からモデル 3 が最も性能が良いことが分かった．また，モデル 1 とモデル 2 に性能の差はあまり現れなかった．

3.2.9　考察

実験結果から，盤面情報から学習が行われていることが確認できた．モデル 1 とモデル 2 に性能の差があまり現れなかったことから，パディングのありなしは学習に影響がないように思われ，モデル 3 が最も性能が良いことからフィルタサイズ 2×2 のものを学習に用いると良いと思われるため，モデル 3 を元にしたモデルを作成することにする．

第 4 章

実験 1:評価に駒色情報を与えない場合

実際のプレイでは相手の情報は分からないため，相手と自分の残り駒数のみ分かっている状況を考え，駒色情報を与えない場合で実験を行い，残り駒数の情報を与えた場合と与えない場合について性能を比較した．

4.1 CNN の構成

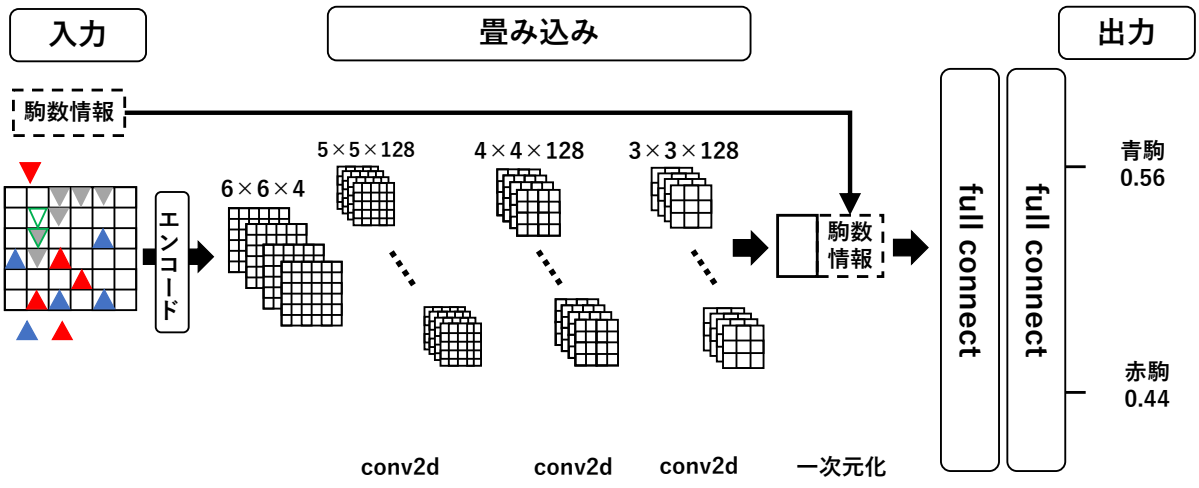

図 4.1　実験 1 の CNN の構成

表 4.1　学習モデルの一致率

学習	評価	一致率
駒数情報なし	駒数情報なし	0.90455
駒数情報あり	駒数情報あり	0.92515

図 4.1 は，実験 1 で用いる CNN の構成である．ネットワークの変更点として，入力データの 4 つの情報に加え自分と相手の残り駒数情報を与える．この駒数情報は，自分の青駒，赤駒と相手の青駒，赤駒の 16 個について 2 値の配列で表したものである．畳み込み層の出力に駒数情報を結合させ, 一次元化を行った後に全結合層に入力する．

50 万局面をバッチサイズ 1000 ごとに 20 エポック学習させ，学習モデルに対して評価を行った．

4.2　結果

実験結果を表 4.1 に示す．実験結果は，どちらも 90%以上の性能が出ていることが分かる．また，残り駒数情報を与えている方が性能が良い．

4.3　考察

実験結果から，残り駒数情報を与えない方よりも残り駒数情報を与えて学習を行うことが有効的だと言える．これは，相手駒の残り駒数から赤駒と青駒がいくつ含まれているかという情報を得ることができ，色推定を行う際の手助けになっているからと考えられる．よって，駒色推定を行う際には盤面情報だけでなく推定の手助けとなる情報を与えることが必要だと考えられる．そのため，残り駒数情報を与えた場合のモデルに対し，追加情報を与えて性能の向上を評価していく．

第 5 章

実験 2:評価に駒色情報を与えた場合

相手の隠された情報が分かることは試合を有利に進めるために重要であり，事前知識として相手の情報が分かっている状況を考える．相手の駒色情報を与えることで推定に有効的であると考えられるため，学習時に駒色情報をどの程度与えると結果が変化するのか，性能の比較を行う．

5.1　CNN の構成

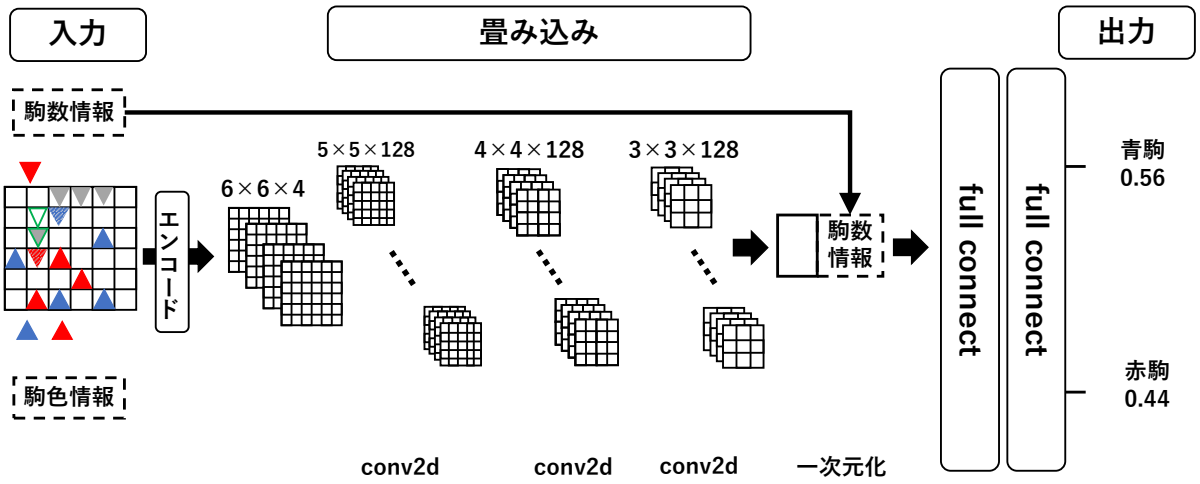

図 5.1　実験 2 の CNN の構成

図 5.2 は，実験 2 で用いる CNN の構成である．

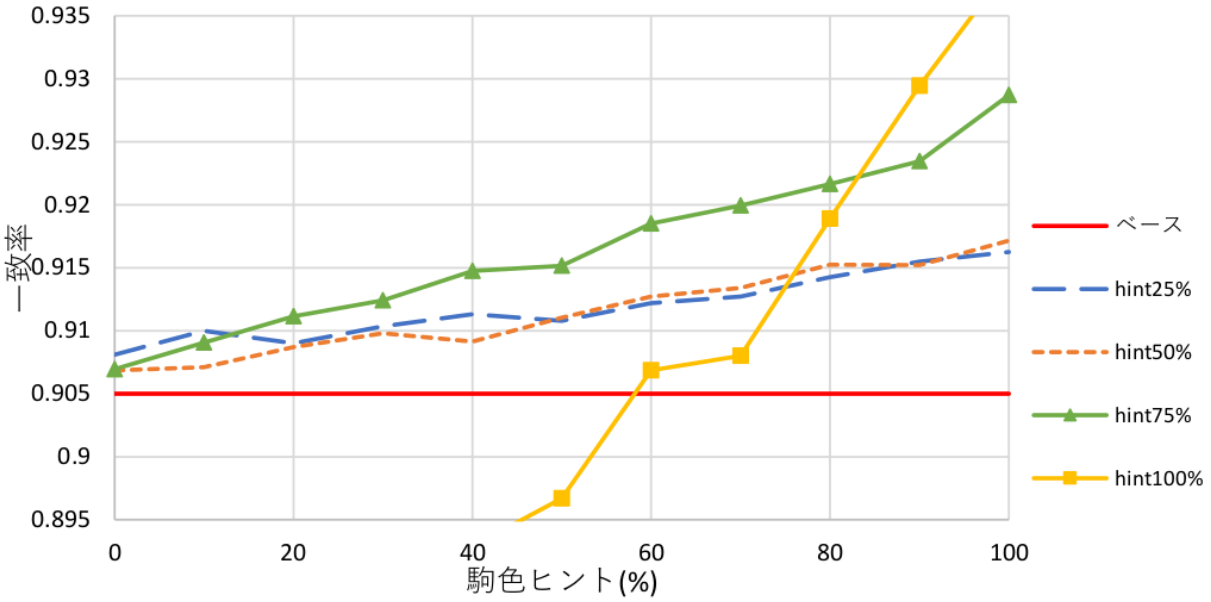

図 5.2　学習に駒色情報を与えた場合の実験結果

ネットワークの変更点として，相手駒の位置の駒色情報を追加情報として与えている．駒色情報は，盤面の縦 × 横 (6×6) と，相手駒の位置，自分の青駒の位置，自分の赤駒の位置，推定する相手駒の位置の情報に追加し 6×6×5 の形としてエンコードを行う．

駒色情報はそれぞれ 0%から 100%の間で 25%刻みに与える．青駒と思われる場所では正の値を割り振り，赤駒と思われる場所では負の値を割り振る．学習したモデルに対して結果がどのように変化するのか評価する．

5.2　結果

図 5.2 は実験結果の図を表している．凡例について，「hint25%」とは学習時に駒色情報を 25%与えているものであることを表しており，以下の「hint50%」,「hint75%」,「hint100%」も同様に駒色情報を数%分与えたものを表している．また，横軸は評価時に駒色情報を全く与えないものから 100%与えたものについてになっている．ベースラインは，学習時も評価時も駒色情報を与えていない場合をベースにしている．実験結果は，図 5.2 から残り駒数と駒色情報を与えたモデルの性能が高く，駒色情報のみのモデルも性能が高いことがわか

る．また，情報の割合で見ると駒色情報のみでの学習では 25%と 50%の性能がほぼ変わらず 75%のとき性能が高いが 100%のときでは性能は良くない．

5.3 考察

実験結果から，追加情報について比較すると残り駒数の情報を持つモデルの方が情報を多く持っており，学習したプレイヤの性能も高くなっていると言える．また，与える情報の割合についてモデルの性能は，情報が多いほど性能も高くなり良くなっているように思われるが，100%のところでは性能が良くない．これは，学習時に情報を与え過ぎて学習すると与えた情報に依存してしまい，情報がなくなるとたちまち性能が落ちるということが言える．

第 6 章

実験 3:学習に間違った情報を与えた場合

実際のプレイでは，通常プレイヤの相手の駒色の推定が 100%正確であるとは必ずしも言えない．色推定が間違ったままゲームを進めると，青駒だと思ったまま相手の駒を取るとすべての赤駒を取ることに繋がり．ゲームの勝率に大きく影響があると考えられる．そのため，学習時に事前情報として間違った情報を含むときと含まないときとで結果がどのように変化するのか評価するために実験を行った．

6.1 CNN の構成

図 6.2 は，実験 1 で用いる CNN の構成である．ネットワークの変更点として，相手駒の位置の駒色情報に間違った情報を追加情報として与えている．

残り駒数と駒色情報を 25%与えたものに対して，間違った情報を全く含まないものと 10%含めたもので学習し，学習したモデルに対して評価時に間違った情報を含めない場合と含めた場合について評価する．

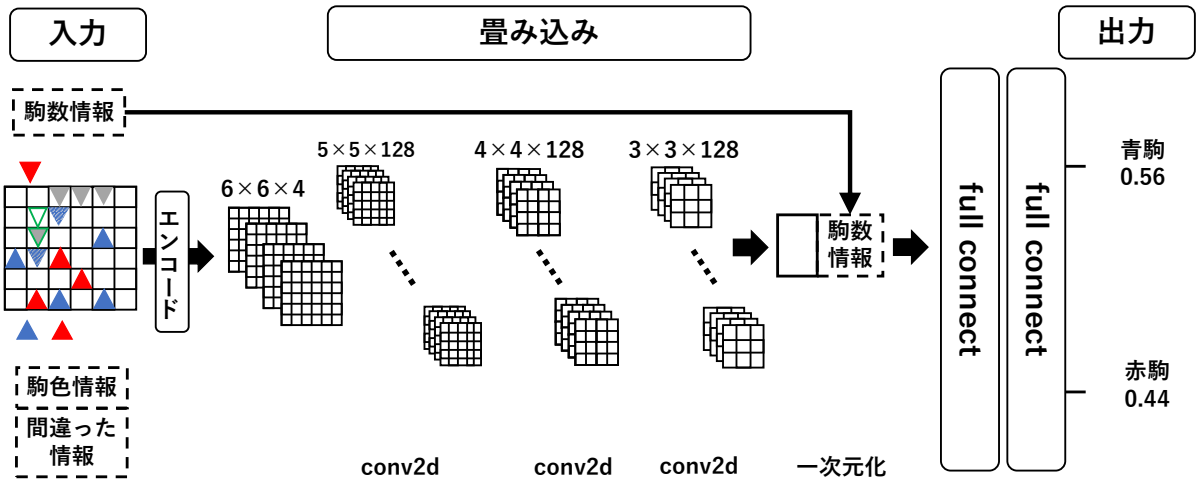

図 6.1　実験 3 の CNN の構成

6.2　結果

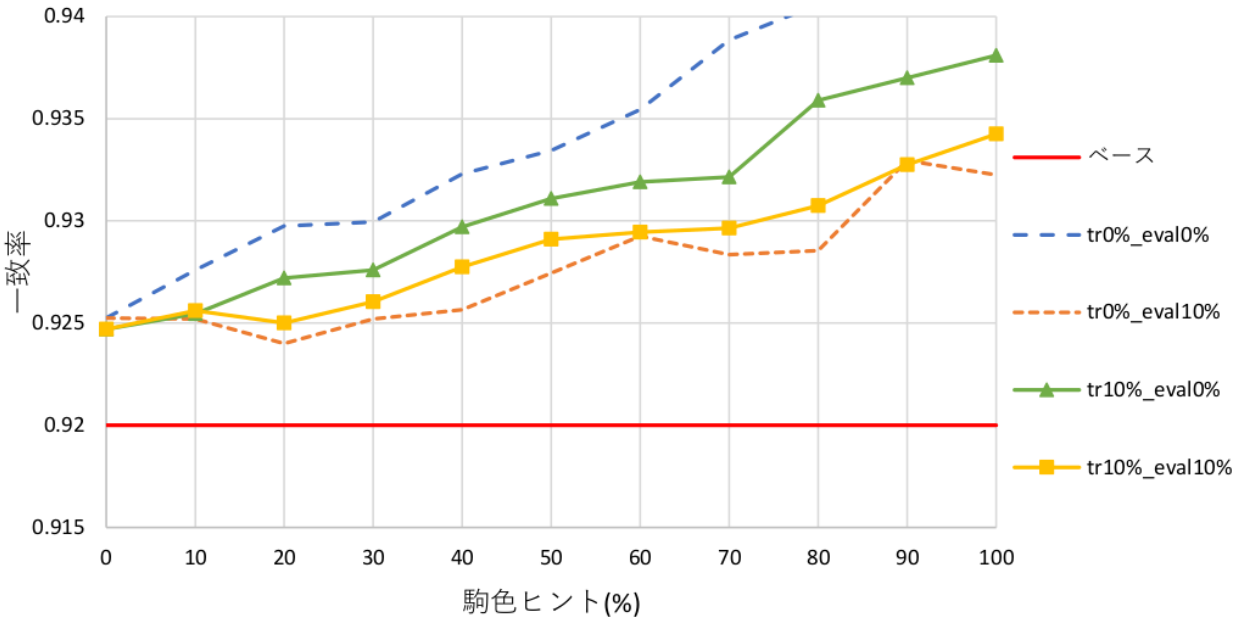

図 6.2　学習に間違った情報を与えた場合の実験結果

図 6.2 は実験 3 の結果を表している．凡例について,「tr」と「eval」はそれぞれ「学習」と「評価」ということを表している．また,「0%」と「10%」はそれぞれ，間違った情報を含めていない場合と間違った情報を 10%含めた場合であることを表している．実験結果から，学習と評価に間違った情報を含めなかった場合の性能が最も良いことが分かり，学習に間違った情報を含めた場合のものの性能も良いことが分かった．また，間違った情報を含めずに学習させたものは含めて学習させたものよりも性能が下がった．

6.3　考察

実験結果から，学習において間違った情報を含めずに学習すると，間違った情報に騙されてしまっていることが分かる．その一方で，間違ったの情報を含めて学習させた場合では性能が向上していることから，学習時に間違った情報を学習することで間違った情報が混ざる状況下でも間違った情報に騙されることなく，相手の駒色を正しく推定ができたのではないかと考える．

第 7 章

関連研究

2017 年からガイスター AI の大会が開催されており [2]，ガイスターの AI 研究は盛んに行われいる．ガイスターにおける相手の駒色を推定する研究について関連研究を示していく．

栃川 [5] らは，ガイスターにおける相手駒色推定の有効性について，相手駒の色推定を行うプレイヤの推定駒数による勝率の変化について評価実験を行っている．実験では，初期盤面から相手の駒色がいくつか分かっている状況でモンテカルロプレイヤで対戦を行い，勝率の変化を調査している．実験結果から，推定する駒数が多いほど勝利率が良くなり，勝率は 90%に収束することが分かっている．

相手駒色推定を行う方法では，モンテカルロ法を用いたものとヒューリスティックを用いたものがある．三塩 [3] らは，モンテカルロ法を用いた駒色推定を行うアルゴリズム UPP を提案している．これは推定したい駒色を青，赤どちらかを仮定し ，モンテカルロシミュレーションで，推定している駒の手の勝率の高さによって色推定を行っている．また鴛淵 [4] らは，アルゴリズム UPP の実際の効果について評価実験を行っており，アルゴリズム UPP のプレイヤは原始モンテカルロプレイヤに対して駒色推定の一致率が 80%で勝率が 56%であると示している．

またヒューリスティックを用いたものでは，ガイスター AI 大会で使用されているプレイヤの多くはヒューリスティックによる駒色推定を行うものが多い．末續ら [7] らは，相手の指す手の振る舞いから相手駒の“ 青らしさ ”を 7 つの評価項目と駒の位置から評価値の更新を行い駒色推定を行っている．このプレイヤはガイスター AI 大会である GAT2018 や GPW2018 および GPW2019 において優勝している．

第 8 章

まとめ

本研究では，CNN を用いたアプローチで相手の駒色を推定する実験を行った．モンテカルロ法ベースのプレイヤのプレイ履歴を学習と評価として使用した．学習モデルを元にして残り駒数と相手の駒色の情報と間違った情報を加えたモデルを作成し学習と評価実験を行った．実験結果より，学習モデルに駒数情報を加えることで性能を向上させることができ，駒色情報を入れるとさらなる性能の向上が見られた．また，相手の駒色情報に間違った情報を含めずに学習を行うと駒色の間違った情報に騙されやすくなり，含めて学習すると間違った情報が混ざる状況下でも騙されることなく正しく推定ができるようになることが分かった．

謝辞

本研究の遂行にあたり，指導教員である松崎公紀教授には研究活動や論文執筆など多くのご指導を頂きました．心より感謝を申し上げます．また，高田喜朗准教授，竹内聖悟講師には副査を引き受けていただき心より感謝を申し上げます.

参考文献

[1] Geister, http://www.luding.org/cgi-bin/GameData.py/ENgameid/19686 (2013).

[2] ガイスター AI 大会, http://www2.matsue-ct.ac.jp/home/hashimoto/geister/GPW/2017/index.html(2017).

[3] 三塩武徳, 小谷善行: ゲームの不完全情報推定アルゴリズム UPP とそのガイスターへの応用, 第 31 回ゲーム情報学研究会 (2014).

[4] 鴛淵 隆斗, 佐藤 直之: ガイスターゲームにおけるモンテカルロ法を利用した駒推定およびブラフ手の生成可能性の検証, 第 43 回ゲーム情報学研究会 (2020).

[5] 栃川純平, 竹内聖悟: ガイスターの初期盤面における相手駒推定の有効性, 第 25 回ゲームプログラミングワークショップ (20)

[6] 木村勇太, 伊藤毅志: 深層強化学習を用いたガイスター AI の構築, 第 24 回ゲームプログラミングワークショップ (2019)

[7] 末續鴻輝, 織田祐輔: ガイスターにおける必勝局面を利用した敵駒推定, 第 43 回ゲーム情報学研究会 (2020).