どうぶつしょうぎを用いたAlphaZeroの手法の調査

中屋敷 太一[1,a)]　金子 知適[2,b)]

概要：AlphaZero は同一のアルゴリズムで強いプレイヤを作成できることを将棋，チェス，そして囲碁の 3 つのゲームのそれぞれで示した．しかし AlphaZero の手法は，どのくらいの学習でどのくらい強くなるかなどを理論的に解析することは難しく，プレイヤ強さを測るには実験的に行うしかない．本稿では AlphaZero の手法で学習を行ったニューラルネットワークがどの程度正しい判断をしているかを，すでに完全解析されたゲームであるどうぶつしょうぎを用いて，完全解析結果と比較し測定した．また異なる大きさのニューラルネットワークを用いて実験を行い，ニューラルネットワークの大きさによる影響を測定した．さらに完全解析結果を用いた教師あり学習も行い，ニューラルネットワークの大きさそのものによる性能比較も行った．最後に AlphaZero が指し手決定の際に用いている探索アルゴリズムである Monte-Carlo Tree Search について，そのハイパーパラメータによる違いを簡単に調査した．実験の結果，教師あり学習の場合には大きいニューラルネットワークほどよい性能である一方で，AlphaZero の手法で用いる際には必ずしもそうではないことを示した．また Monte-Carlo Tree Search のハイパーパラメータによって探索の挙動が大きく変わることを示した．

A Survey on AlphaZero Algorithm through Dobutsu Shogi

Taichi Nakayashiki[1,a)]　Tomoyuki Kaneko[2,b)]

Abstract: AlphaZero succeeded to make a strong player with its alrogithm on each game of Shogi (Japanese chess), Chess and Go. However, it is a hard work to analyze the relationship between learning amount and strength of AlphaZero theoretically, so experiments are needed to measure its strength. In this paper, we investigate performance of neural networks which trained in AlphaZero algorithm via Dobutsu Shogi that has already solved, comparing solved data. We trained neural networks of different sizes and compare them. Then we conduct supervised learning on neural networks of several sizes with solved data and compare the difference among them. Finally, using Monte-Carlo Tree Search that is used when AlphaZero decides the next move, we investigate effects of its hyper parameter. As a result, we found that larger neural networks have better performance in supervised learning of our experiments. On the other hand, larger neural networks can be worse in AlphaZero algorithm. Subsequently, we found that the hyper parameter is not negligible for its behavior.

1. はじめに

近年，将棋，チェス，そして囲碁におけるコンピュータプレイヤは非常に強くなっている．また DeepMind 社が開発した AlphaZero は，同一のアルゴリズムでの強いプレイヤの作成に将棋，チェス，そして囲碁 3 つのゲームでそれぞれ成功し，当時のトップコンピュータプレイヤにいずれのゲームでも勝ち越した [1]．

アルゴリズム，ハードウェアの進歩とともにコンピュータプレイヤが強くなるにつれ，コンピュータプレイヤがどのくらい優れているのかを人間が判断することはもはや困難となっている．そこでトップコンピュータプレイヤどうしを実際に対局させ，Elo レーティングからその強さを判断するという手法が主流である．しかし，この手法ではプ

[1] 東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
[2] 東京大学大学院情報学環
Interfaculty Initiative in Information Studies, the University of Tokyo
[a)] tnakayashiki@g.ecc.u-tokyo.ac.jp
[b)] kaneko@acm.org

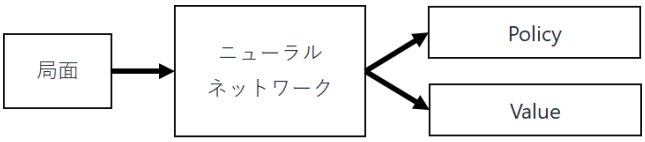

図 1　ニューラルネットワークの概略図
Fig. 1　Overview of neural networks

レイヤが様々な局面でどのくらい正しい判断をできるのかを評価するのは困難である．

AlphaZero の手法は強いプレイヤを作成することに成功したが，AlphaZero の手法がどのくらいの学習でどのくらいの強さになるのかを理論的に解析することは難しい．そのため AlphaZero の手法によって得られるプレイヤの強さは実験的に測定する必要がある．本稿では完全解析されたどうぶつしょうぎを用いて，AlphaZero の手法によって得られるプレイヤの判断の定量的な評価を行った．完全解析されたゲームで実験することで，AlphaZero の手法でどのくらい正しい判断ができるようになるのか，使用するニューラルネットワークの大きさによって正しい判断の割合がどのくらい変わるのか，を定量的に測定することができる．また本稿では完全解析データベースを用いた教師あり学習も行い，ニューラルネットワークの大きさそのものによる性能比較も行った．

2. 先行研究

2.1 AlphaZero

AlphaZero [1] は DeepMind 社によって AlphaGo [2], AlphaGo Zero [3] に続いて開発された．AlphaGo と AlphaGo Zero では囲碁における強いプレイヤを作成することを目標とし，成功した．AlphaGo Zero では既存の棋譜を使用することなく自己対戦のみで強いプレイヤを作成することに成功した．AlphaZero では AlphaGo Zero と殆ど同じアルゴリズムで将棋，チェスにおいても強いプログラムを作成することに成功した．

AlphaZero の手法は大きく，自己対戦による教師生成と，ニューラルネットワークの学習に分けることができる．

2.1.1 自己対戦による教師データの生成

AlphaZero の手法では，自己対戦によって生成された棋譜を教師として局面評価に用いるニューラルネットワークの学習を行う．AlphaZero に用いられたニューラルネットワークは，局面を入力として受け取り，次の手の確率分布 $\boldsymbol{p}$ (Policy) と評価値 v (Value) を出力する (図 1)．

自己対戦を行う際には探索アルゴリズムである Monte-Carlo Tree Search (MCTS) が用いられている．MCTS にはいくつかの種類があり細部が異なるが，ここでは AlphaZero の手法で使われたものについて記述する．

MCTS が用いるゲーム木の各ノード s は $\{N(s), Q(s), W(s), P(s)\}$ の 4 つの値を持ち，それぞれ 0 を初期値として持つ．$N(s)$ はノードの探索回数，$Q(s)$ は s 以下のノードの評価値の平均，$W(s)$ は s 以下のノードの評価値の和，$P(s)$ は親ノードから s への遷移確率を表す．MCTS は選択，展開，評価，そして逆伝播の 4 ステップからなり，これらをまとめてシミュレーションと呼ぶ (図 2)．

(1) 選択
根ノードから，各ノード s で式 (1) の PUCT の値が最大となる s の子 c を再帰的に選び末端の子を選ぶ．

$$\mathrm{PUCT} = Q(c) + CP(c)\frac{\sqrt{N(s)}}{1+N(c)} \tag{1}$$

C は定数で，有力そうな子ノードを更に探索するか，まだあまり探索できてないノードを探索するかを調整する．
ここで選ばれた末端ノードを s_L とする．

(2) 展開
s_L での合法手を列挙し，各合法手で遷移する先の局面を s_L の子ノードとして追加する．

(3) 評価
s_L の局面をニューラルネットワークで評価し，$\boldsymbol{p}, v$ を得る．s_L の局面での合法手の中で $\boldsymbol{p}$ を Softmax 関数を用いて正規化し (これを $\boldsymbol{p^*}$ とする)，s_L の各子ノード c' に対して $P(c') = p^*_c$ とする．

(4) 逆伝播
s_L で得られたニューラルネットワークの評価値 v を，末端ノードから親ノードに向かって伝播する．s_L から根ノードの間の各ノード s に対して，$N(s) = N(s)+1$, $W(s) = W(s) + v$, $Q(s) = W(s)/N(s)$ と更新する．このとき v は一つノードを遡るごとに符号を反転させる．

MCTS は固定時間または固定回数シミュレーションを行い，最終的に根ノードの各子の訪問回数の頻度分布 $\boldsymbol{\pi}$ を出力する．

自己対戦では生成される棋譜の多様性のため，根ノードの場合には評価のステップで $\boldsymbol{p}^*$ に以下のように Dirichlet ノイズが加えられている．

$$\boldsymbol{p}^* \leftarrow (1-\varepsilon)\boldsymbol{p}^* + \varepsilon \mathrm{Dir}(\alpha)$$

ここで ε, α は定数である．

ゲームのルールによって先手勝ち，後手勝ち，または引き分けが決まると，その対局を終了し，教師データとして各手番の MCTS の出力 $\boldsymbol{\pi}$ と対局結果を格納する．これを繰り返し教師データを作成し続ける．

2.1.2 ニューラルネットワークの学習

ニューラルネットワークの出力 Policy, Value に対しそれぞれ各局面の MCTS の出力 $\boldsymbol{\pi}$，各局面の対局結果 z ($z=1$ 手番の勝ち，$z=0$ 引き分け，$z=-1$ 手番の負け) を教師

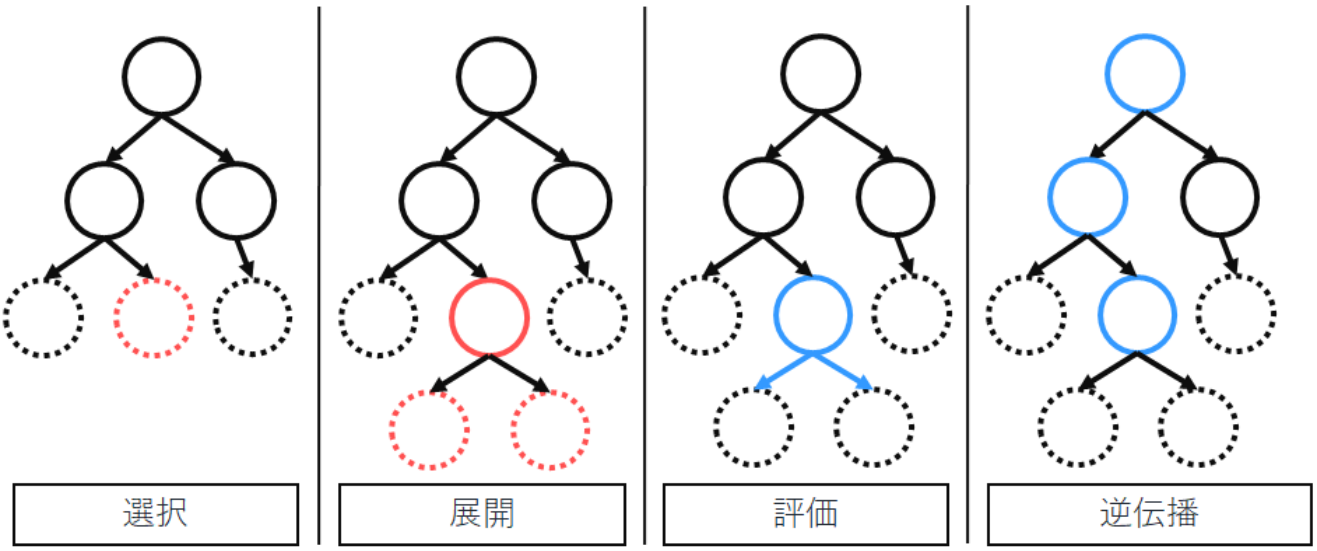

図 2　MCTS の 4 ステップ．点線で描かれているノードはまだ評価されていない末端ノード．
Fig. 2　Four steps of MCTS. Dotted line denotes leaf nodes that are not evaluated yet.

表 1　どうぶつしょうぎの駒の動かし方
Table 1　Movements of each piece

駒	方向
ひよこ	上
きりん	上，右，下，左
ぞう	右上，右下，左下，左上
ライオン	上，右上，右，右下，下，左下，左，左上
にわとり	上，右上，右，下，左，左上

として学習を行う．式 (2) のロス関数を最小化するようにニューラルネットワークの学習を行う．

$$l(\theta) = (z - v)^2 - \boldsymbol{\pi}^{\mathrm{T}} \log \boldsymbol{p} + c||\theta||^2 \qquad (2)$$

ここで θ はニューラルネットワークのパラメータを表す．c は L2 正則化のパラメータであり，10^{-4} を用いる．

2.2　どうぶつしょうぎの完全解析

どうぶつしょうぎとは，本将棋の派生ゲームの一種である．基本的なルールは本将棋とほとんど同じだが，用いる将棋盤の大きさが 4×3 と小さく，駒の種類も 5 種類と少なくなっている．5 種類の駒はひよこ，きりん，ぞう，ライオン，にわとりであり，ひよこは相手陣に行くことで，にわとりになることができる．初期配置は図 3 である．それぞれの駒は表 1 のように 1 マスだけ動く．

どうぶつしょうぎは取りうる状態数が少なく完全解析されており，初期局面から到達可能な全局面数は 246 803 167 局面であり，初期局面では後手勝ちであることが知られている [4].

3.　提案手法

Elo レーティングによるプレイヤの強さの評価では，様々

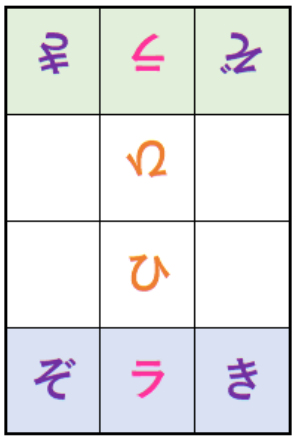

図 3　どうぶつしょうぎの初期配置．「ひ」，「き」，「ぞ」，「ラ」はそれぞれひよこ，きりん，ぞう，ライオンを表す．青色のマスは先手陣，緑色のマスは後手陣を表す．
Fig. 3　The initial position of Dobutu Shogi.

な局面で正しい判断をできているかどうかについては評価できない．そこで全ての局面でゲームの理論的な答えと比較するのは強さを評価するための一つの有力な手法である．本稿では完全解析されているものの強化学習にとっては簡単な問題ではないどうぶつしょうぎを用いて，AlphaZero の手法で学習したニューラルネットワークが，どのくらい正しい判断をできているかどうかを測定する．また異なる大きさのニューラルネットワークを用いて実験を行い，AlphaZero の手法におけるニューラルネットワークの大きさによる影響を測定する．さらに完全解析データベースを教師とした学習も行い，ニューラルネットワークの大きさそのものによる変化を調べる．

本稿では MCTS に関する簡単な調査も行った．

表 2　各チャネルの特徴
Table 2　Features of each channel

特徴	チャネル数
現手番の盤上の駒	5
相手番の盤上の駒	5
現手番の各持ち駒の数	3
相手番の各持ち駒の数	3
先手番かどうか	1
現在の手数	1

表 3　ニューラルネットワークの構造.
角括弧は各層の出力の形を表す.

Table 3　The structure of neural networks

入力 $[18,4,3]$	
畳み込み $[256,4,3]$ ReLU $[256,4,3]$	
Residual ブロック $[256,4,3]$ Residual ブロック $[256,4,3]$ ⋮ Residual ブロック $[256,4,3]$	
畳み込み $[256,4,3]$ Batch 正規化 $[256,4,3]$ ReLU $[256,4,3]$ 全結合 $[144]$	畳み込み (*) $[1,4,3]$ Batch 正規化 $[1,4,3]$ ReLU $[1,4,3]$ 全結合 $[256]$ 全結合 $[1]$ TanH $[1]$
Policy	Value

表 4　Residual ブロックの構造.
角括弧は各層の出力の形を表す.

Table 4　The structure of residual block

入力 $[256,4,3]$	
	畳み込み $[256,4,3]$ Batch 正規化 $[256,4,3]$ 畳み込み $[256,4,3]$ Batch 正規化 $[256,4,3]$
足し合わせ $[256,4,3]$ ReLU $[256,4,3]$	

AlphaZero では次の指し手を決定する際に MCTS を用いて探索をしている．本稿ではニューラルネットワークの出力がゲームの理論的答えから大きく異なる局面で，MCTS を行うことで理論的答えに近づくのかどうかを調査する．また異なるいくつかのハイパーパラメータで MCTS を行い，MCTS の挙動の違いを調べる．

4. 実験

4.1 ニューラルネットワークの構造

AlphaZero に用いられたニューラルネットワークはサイズが大きく学習に時間がかかってしまう．そこで本稿では簡単にしたニューラルネットワークを用いた．AlphaZero ではニューラルネットワークの入力層に，現在の局面だけではなく過去 8 手分の局面を使用していたが，本稿では現局面のみを使用する．局面は手番のプレイヤの視点で入力層に設定する (後手番の場合には盤面を反転する).

入力層は 18 チャネルの 4×3 ピクセルからなる．各チャネルには表 2 に示す値を設定した．ニューラルネットワークの構造を表 3 に示す．畳み込みのフィルターサイズは 256 であり，カーネルサイズは $[3,3]$ である．なお (*) の畳み込みではフィルターサイズが 1，カーネルサイズが $[1,1]$ のものを用いた．Residual ブロック [5] は表 4 の構造をしており，入力と，それを 2 つの畳み込み層で処理した結果を足し合わせる構造をしている．

Policy は 144 個のスカラー値を出力する．盤上の駒を移動する指し手を，移動方向 (8 方向)，移動元の場所 (4×3) の $8\times4\times3$ 個のスカラー値で表す．持ち駒を打つ指し手を，持ち駒の種類 (3 種類)，打つ場所 (4×3) の $3\times4\times3$ 個のスカラー値で表す．ひよこが成る手を，移動元のひよこの場所の 4×3 のスカラー値で表す．以上合計 144 個のスカラー値で全合法手を表現することができる．

4.2 異なる大きさのニューラルネットワークの性能の比較

ニューラルネットワークの大きさは強さに大きく影響すると考えられる．AlphaZero では 19 個の Residual ブロックを用いたニューラルネットワークを使用している．

本稿では，Residual ブロックによる性能差を測定するため，異なる Residual ブロックの数のニューラルネットワークを用いてそれぞれ AlphaZero の手法による学習を行った．またニューラルネットワークの大きさそのものによる性能差を比較するため完全解析データベースを用いた教師あり学習も行った．教師あり学習では Residual ブロック 1 個，3 個および 5 個を用いたネットワークを，AlphaZero の手法では 3 個および 5 個を用いたネットワークの学習を行った．学習の詳細については付録 A.1 に記載した．

はじめに Value の出力に関する正答率の測定について記述する．ニューラルネットワークの Value の出力 v は $v\in(-1,1)$ であるため，引き分けの局面の処理は簡単ではない．そのため，本稿では勝ち局面と負け局面だけを対象として測定した．Value の出力 v と実際の対局結果 z の符号が等しいときに正答として正答率を測定した．

次に Policy の出力に関する正答率の測定について記述する．ある局面が勝ち局面ということは，その局面の合法手の中に必ず勝ち局面に遷移する手が含まれている．そこで，与えられた局面における全ての合法手 $\mathcal{A}$ の中で Policy の出力が最大のもの，すなわち $\arg\max_{a\in\mathcal{A}} p_a$ で遷移する局面が勝ち局面かどうかで Policy の正答率を測定した．

Value の正答率を測定する際には非末端局面の勝ち局面 56 474 473 局面および負け局面 40 328 395 局面を用いた．Policy の正答率を測定する際には非末端局面の勝ち局面を

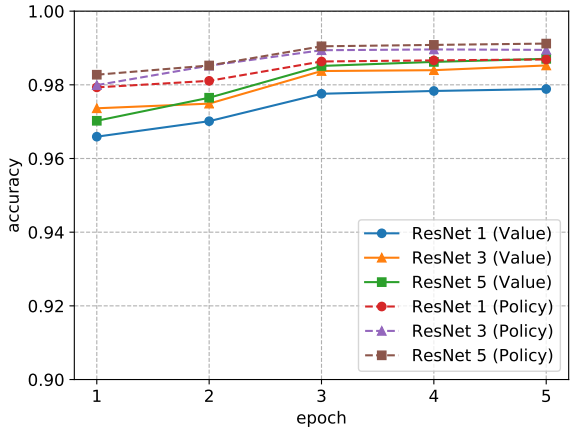

図 4　異なる Residual ブロックの数での正答率の比較 (教師あり学習)

Fig. 4　Accuracy on neural networks with different number of residual blocks (supervised learning)

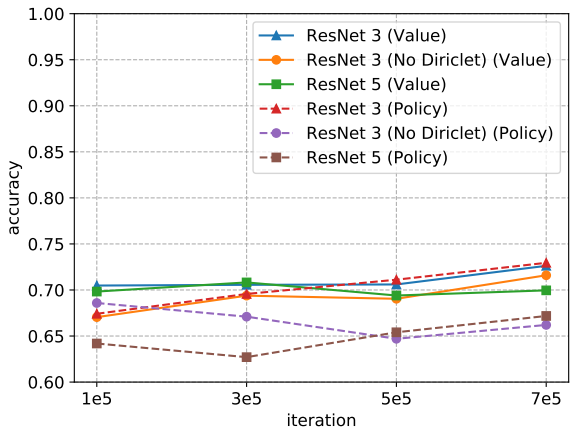

図 5　異なる Residual ブロックの数での正答率の比較 (AlphaZero の手法)

Fig. 5　Accuracy on neural networks with different number of residual blocks (AlphaZero algorithm)

用いた．

以上の測定方法による，教師あり学習の結果を図 4 に，AlphaZero の手法での学習の結果を図 5 に示す．

教師あり学習の結果から Residual ブロックの数が多いほど正答率が高いことがわかる．そのため，教師データの質が良ければ大きいニューラルネットワークを使った場合のほうが性能が良いと考えられる．しかし一方で，AlphaZero の手法では Residual ブロックが 5 個の場合の正答率より，3 個の場合の正答率が高いことがわかる．これは図 4 で Residual ブロック 3 個のものと 5 個のものを比較してわかるように，大きいニューラルネットワークの学習には時間がかかることに起因すると考えられる．そのため学習の始めのうちは小さいニューラルネットワークを用いて学習を行い，棋譜の質が高くなるにつれて大きいニューラルネットワークを使うことで，学習の効率向上に繋がると考えられる．

4.3　MCTS による性能向上

AlphaZero の手法では，ニューラルネットワークの出力をそのまま使うのではなく，MCTS を行いその結果に従って次の指し手が決定される．MCTS を行うことで，ニューラルネットワークの出力のみよりも性能が向上すると考えられている．これを検証するため，MCTS によって局面評価が正確になるのかどうかについて簡単な調査を行った．またハイパーパラメータによる違いについても調査した．ニューラルネットワークは，AlphaZero の手法で学習した Residual ブロック 3 個のものを用いた．

解析結果は勝ちだが，ニューラルネットワークによる Value の出力が $v \approx -1$ である局面 (図 6) を対象の局面として調査した．

AlphaZero では式 (1) の C に，定数ではなく式 (3) で表される値を用いている．

$$C = \log\left(\frac{N(s) + 19652 + 1}{19652}\right) + 1.25 \qquad (3)$$

探索が進むにつれてこのように C を調節することで，まだあまり探索できていない局面をより探索するようになる効果がある．本稿では式 (1) の C に式 (3) を用いた場合，$C = 1.0, 1.5, 2.0$ の場合それぞれで MCTS の挙動がどのように変わるかを調査した．MCTS のシミュレーション回数に沿った局面 s の評価値 $Q(s)$ の変化を図 7 に示す．

図 7 の結果からこの局面ではいずれの場合にも，MCTS を使うことで理論的な答えに近づいていることがわかる．また $C = 1.0$ とした場合が一番理論的答えに近いことがわかる．これは，理論的に勝ちの局面だが勝ちとなる正しい指し手が 1 手しかないような局面の場合に，C を大きくすることで様々な局面を探索し，その結果評価値の平均である $Q(s)$ が下がるためであると考えられる．この C は探索性能に大きく影響を与えると考えられ，そのため，もしこの C を局面に合わせて調整できればより強いプレイヤになると考えられる．

5.　まとめと今後の課題

本稿では完全解析されたゲームであるどうぶつしょうぎを用いて，AlphaZero の学習手法のニューラルネットワークの性能を定量的な測定を行った．また完全解析データベースを用いた学習も行うことで，ニューラルネットワークの大きさそのものの性能比較と，AlphaZero の手法に用いた際の性能比較を行った．

実験では教師棋譜の質が良いときにはニューラルネットワークの大きさが大きい方が良いが，AlphaZero の手法で用いるニューラルネットワークは必ずしも大きい方が良いわけではないことを示した．また実際に MCTS を用いた

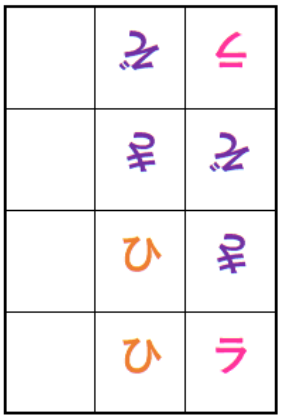

図 6　ニューラルネットワークの出力が理論的な答えと大きく異なる局面の例 (先手番)

Fig. 6　An example position that the value output is much different from the theoretical value

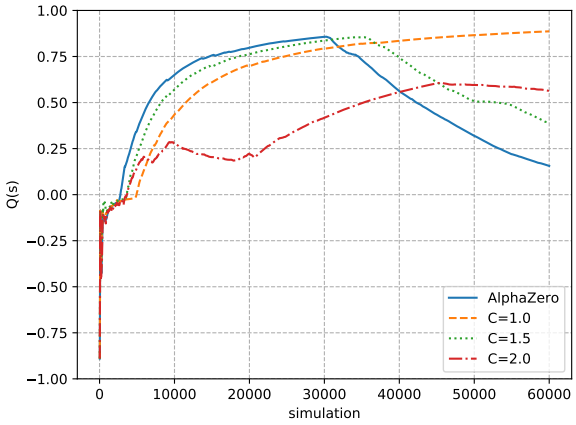

図 7　MCTS の C による評価値の違い

Fig. 7　Q value of different C in MCTS

際の評価値の変化を調べ，MCTS が有効であることを示した．さらに MCTS ハイパーパラメータによって得られる結果に少なくない差が生まれることを示した．

AlphaZero の手法では引き分け局面についての処理は簡単ではないため本稿では扱わなかった．今後の課題として引き分け局面についても評価の指標が導入できればと思う．また MCTS での C を局面ごとに調整することでより強いプレイヤの作成ができるのではないかと考えられる．

参考文献

[1] 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).

[2] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. and Hassabis, D.: Mastering the game of Go with deep neural networks and tree search, *Nature*, Vol. 529, pp. 484–503 (2016).

[3] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T. and Hassabis, D.: Mastering the game of Go without human knowledge, *Nature*, Vol. 550, pp. 354– (2017).

[4] 田中哲朗：「どうぶつしょうぎ」 の完全解析，研究報告ゲーム情報学（GI），Vol. 2009, No. 3, pp. 1–8 (2009).

[5] He, K. et al.: Deep Residual Learning for Image Recognition, *CoRR*, Vol. abs/1512.03385 (2015).

[6] Taichi, N. and Tomoyuki, K.: Learning of Evaluation Functions via Self-Play Enhanced by Checkmate Search, pp. 126–131 (online), DOI: 10.1109/TAAI.2018.00036 (2018).

[7] Chaslot, G. M. J. B. et al.: Parallel Monte-Carlo Tree Search, *Computers and Games*, Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 60–71 (2008).

付　　録

A.1　学習の設定

ニューラルネットワークの学習にはモーメント付き SGD を用いた．モーメントの値は 0.9 とし，バッチサイズは 2048 とした．学習率は 0.01 から始め，100 000，300 000，500 000 イテレーションで 0.1 倍した．ここでニューラルネットワークのパラメータを 1 回更新することを 1 イテレーションと呼ぶ．

A.1.1　教師あり学習

GeForce GTX 1080Ti 1 基を使用しニューラルネットワークの学習を行った．教師データとして非末端の局面を使用した．なおデータベースでは反転して同一局面になる局面については一つのエントリとして登録されてあるため，学習の教師として設定する際に，確率 1/2 で盤面を反転したものを使用した．

教師あり学習は完全解析データベースを用いて 5 エポック行った．ここではデータベースに収録されている全局面を 1 回ずつ 1 度学習することを 1 エポックとする．1 エポックは 48576 イテレーションに相当する．学習には Residual ブロック 1 個のネットワークでは約 10 時間，3 個のネットワークでは約 15 時間，5 個のネットワークでは約 20 時間を要した．

A.1.2　AlphaZero の手法

ニューラルネットワークの学習と自己対戦それぞれに GeForce GTX 1080Ti 1 基ずつ使用した．自己対戦では 5 つの対局を並列で行った．ニューラルネットワークの学習対象の棋譜として直近に生成された 10 000 棋譜から一様ランダムに 2 048 局面選んだ．MCTS での Dirichlet ノイズを加える際には $\varepsilon = 0.25, \alpha = 0.34$ を用いた．

学習を高速化するため，MCTS の根局面では 7 手の詰将棋探索を行い，詰みが見つかった場合にはその手を選ぶようにした [6]．その局面を教師として用いる際には，Policy の教師としてその手のみが 1 の one-hot ベクトルを用いた．1 局の中で同じ局面が 2 度目に現れたときには引き分けとした．

Residual ブロック 3 個のニューラルネットワークの学習の際には約 85000 棋譜が生成され，学習時間は約 3.5 日であった．Residual ブロック 5 個のニューラルネットワークの学習の際には約 80000 棋譜が生成され，学習時間は約 4.5 日であった．学習 1 イテレーションあたりに生成される棋譜の数は，この 2 つのニューラルネットワークの間で殆ど変わらなかった．

学習時のロスを図 A·1 に示す．また学習に伴う初期局面の Value の出力の値を図 A·2 に示す．

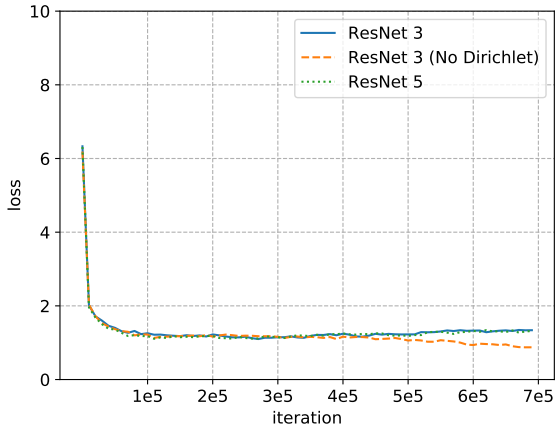

図 **A·1**　学習時のロス

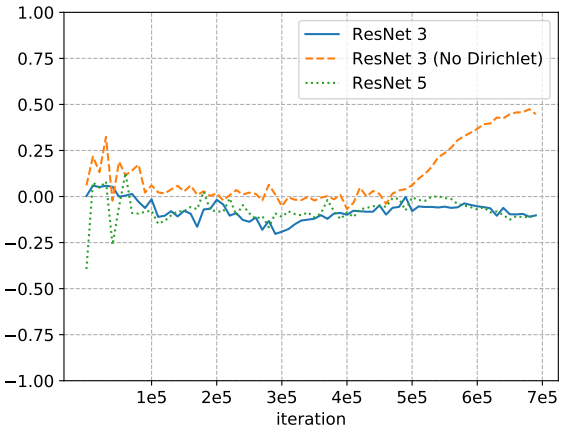

図 **A·2**　初期局面での Value の出力

A.2　MCTS での局面評価のバッチ処理

MCTS では Virtual loss [7] を用いて 32 局面を一度に選択し，これをまとめてニューラルネットワークで評価した．

Virtual loss は MCTS を並列化した際に，MCTS の選択のステップで使用される手法である．複数スレッドが同時に選択のステップを実行すると，同じ末端ノードにたどり着いてしまい無駄となってしまう．これを回避するため，他のスレッドが選択のステップで選んだノードを，逆伝播が行われるまで，1 スレッドにつき 1 回負けたとみなすという手法である．

本稿では複数スレッドで探索を行うのではなく，単一スレッドで選択を行うが選択のステップを 1 度に 32 回行い，32 個の末端ノードを集める実装とした．なおこの実装でも GPU によるニューラルネットワークの実行がバッチ処理と成るため十分な高速化となる．

A.3 Dirichlet ノイズの影響

AlphaZero では自己対戦によって得られる棋譜に多様性を与えるため，Policy の出力に Dirichlet ノイズを加えている．本稿では AlphaZero の手法で Dirichlet ノイズを用いない学習も行い，このノイズによる差異を調査した．図 5，図 A·1，図 A·2 に結果を示す．

図 5 から正答率に関しては Dirichlet ノイズを用いたものと同程度である．しかし，ロスが Dirichlet ノイズを用いたものと比べ小さい値になっており，また初期局面での Value の出力が先手勝ちの評価に近づいていることがわかる．これは Dirichlet ノイズが無いため同じ棋譜を生成することが増え，過学習を起こしていると考えられる．そのため今後さらに学習を続けた際に Dirichlet ノイズを用いたものに比べ性能が悪くなることが考えられる．