深層学習を用いた囲碁における悪手の評価について

佐藤 慎也[1,a)]　山本 修身[1]

概要：本稿では，囲碁における良手と悪手の識別のために，深層学習を用いて識別機を構築する手法およびそれを囲碁プログラムへ応用した場合の効果について検証した．過去のゲームの記録（棋譜）における着手点を良手，それ以外の座標をすべて悪手と仮定し，多数の 9 路盤における棋譜を用いて機械学習を試みた．その際，ニューラルネットワーク（NN）を用い，3 層の NN を用いた識別機および NN の一種である深層学習による識別を試みた．その結果，テストデータにおける良手ラベルの識別で 99.5%，悪手ラベルの識別で 51.6%で正しく識別を行うことができた．また，この識別機を用いたモンテカルロ木探索の枝刈りを行うプログラムを実装した．このプログラムで枝刈りしないプログラムと対局を行った時，8%勝率が向上した．この結果から，本稿の識別機が判別した悪手の枝刈りは効果があることを確認できた．

On Detection of Bad Moves for the Go game Using Deep Learning

Shinya Sato[1,a)]　OsamiYamamoto[1]

Abstract: This paper describes a method for constructing a classifier which distinguishes good and bad moves in Go games and an evaluation of the performance of the classifier. Assuming that actual moves in recorded scores of Go games are good ones and others are bad ones, we constructed millions of examples from the recorded scores. We adopted a 3-layered neural network (NN) and a deep learning neural network for learning machines. Using those examples, the deep learning neural network detected 99.5moves as “bad moves.” We, moreover, implemented the classifier in a computer Go program for branch pruning. The winning ratio of the program against a program without the branch pruning improved by eight percent in average.

1. はじめに

古くから世界で親しまれてきた戦略性の高いゲームの一つに囲碁がある．囲碁では，9 × 9 や 19 × 19 の線が引かれた盤面に白と黒の碁石を交互に配置する．プレイヤーは，地と呼ばれる陣地を作り相手より多くの陣地を作り上げる．相手より多くの地を作りあげることが囲碁の目的である．

現在，囲碁のプログラムの実力はアマチュアの 6 段程度である．これは，アマチュアでは非常に強いプレイヤーに分類されるが，囲碁のプロ棋士には及ばない．実際に行われた囲碁のプロ棋士とコンピュータ囲碁によるハンディキャップを設けた対局では，3 子でコンピュータが勝利したこともある [*1]．

コンピュータ囲碁には，2 つの大きな課題がある．まず，膨大な状態数である．ある局面から展開される局面数が非常に多いため，通常の探索手法ではうまくいかない．次に，評価関数を構築することが困難な点である．囲碁プログラムの研究では，2006 年にモンテカルロ木探索 (MCTS) [1] と呼ばれる探索手法が提案され，それ以前の手法の囲碁プログラムの実力を著しく向上した．この探索手法は，シュミレーションでゲームが終了した状態を作成し，その結果を考慮することで統計的に局面を評価しながら探索する手法である．

MCTS の利点は，評価関数を構築しなくても局面の評価

[1] 名城大学大学院，理工学研究科情報工学専攻
Meijo University
[a)] 143430012@ccalmini.meijo-u.ac.jp

[*1] 電聖戦の様子についてのネット記事の URL. http://www.nikkei.com/article/DGXMZO84490320X10C15A3000000/

を統計的に行える点である．あくまでも統計的な局面の評価であるため，シミュレーションの質に依存して評価の正しさが決まる．MCTS では，1 回のシミュレーションにかかる時間とシミュレーションの質でトレードオフが発生する．このため，局面の評価の正しさには限界がある．そこで，MCTS 以外の手法が必要になる．

2. 囲碁における機械学習の取り組みの現状と本稿の目的

コンピュータ囲碁では，機械学習を用いた研究が多くある．その多くは MCTS に応用することを目的としている．

モンテカルロ木探索では，ある状態からランダムにゲームを進行させ，ゲームが終了した状態を作成するシミュレーションを行う．これは，playout と呼ばれる．機械学習の取り組みの一つとして，この playout の質の向上を目的とした研究がある [2]．また，過去のゲームの記録 (棋譜) からプロの手を再現することを目的とした研究がある．近年，深層学習の 1 種である畳み込みニューラルネットワーク (CNN) を用いて棋譜の手を行う研究 [3] [4] が注目を集めている．本稿では，深層学習を用いて棋譜の手を再現することを目的とした研究である．本稿と [3] [4] は，用いる学習機が CNN ではなく Autoencoder と呼ばれるニューラルネットワーク (NN) の次元圧縮のアルゴリズムを用いている点で異なる．

NN を用いた悪手の評価については， [5] である程度可能であることがわかる． [5] では，81 ノードで 9×9 の碁盤を表現し，黒石，白石，石が無い座標をそれぞれ表現して入力し，81 ノードでそれぞれの座標が悪手か否かを 0, 1 で表現する識別機を構築した．本稿では，入力および出力のデータ表現を変更した NN についても識別機の検証を行う．

本稿では，9×9 の碁盤で深層学習を用いて良手と悪手の評価を行う識別機を構築する．この識別機を用いて良手と悪手を正しく識別することが目的である．まず良手と悪手の識別をするために過去の棋譜データを用いて機械学習を行う．このとき，NN の一種である深層学習を用いて学習を行う．学習後はある 1 局面を入力して，ある座標が良手と悪手のどちらに分類されるかを出力する．囲碁プログラムへの応用として，学習済みの識別機の出力が悪手のものを木探索で枝刈りした場合の効果について検証する．

3. 本稿で用いる識別機 NN と深層学習について

本稿では，深層学習による悪手の識別の性能を確認するため，3 層の NN で学習した識別機を用いた学習も行う．まず，3 層の NN の識別機の構成について示す．まず，NN のパラメータとして隠れ層ノードの個数がある．隠れ層についてはノード数だけでなく，隠れ層を 1 以上の任意の層数に定義することが可能である．隠れ層の層数およびそれぞれの層のノード数のパラメータによって識別機の性能に違いが見られる．隠れ層のノード数および層の数が増加するほど入力データの特徴量の表現能力が向上するが，増加するほど学習の収束に時間がかかり，過学習が起こるようになる．次に，NN の層と層の間を繋ぐノード間を特定の学習に適した方法で接続することで性能が向上することが知られている．本稿では，隠れ層を 1 層で 50 ノードとして NN を構築した．また，層と層の間を繋ぐノードをすべてのノードに接続するネットワーク (Fully-connected Neural Networks) を用いる．重みの更新方法として誤差逆伝播法 (Backpropagation) を用いる．また，隠れ層および出力層ではシグモイド関数 (sigmoid function) を活性化関数として用いた．

深層学習は NN の一つである．従来の NN と異なる点は，データの特徴量を自動で発見する能力を持った識別機である点である．これにより，効果的な機械学習では必須と考えられていた入力データのテクニカルな特徴抽出をせずとも，識別機が自動で特徴を認識し効果的な識別機を構築することが可能になる．深層学習にさまざまあり，特徴抽出のアルゴリズムと従来の識別機のアルゴリズムの選択次第で幾通りも手法が存在する．本稿では，特徴抽出の手法として SDA [7]，識別機の手法として MLP を用いることで識別機の構築を試みる．

事前学習 (Pre-training) では Autoencoder(AE) [6] や Re-stricted Boltzmann Machine(RBM) などを用いて特徴を抽出する．今回は AE を用いてデータの特徴抽出を行った．AE は入力層，隠れ層，出力層の 3 層ニューラルネットワークで構成される．AE では，入力層と出力層が同じになるように隠れ層までの重みを学習する教師なしの学習を行う．これを実現することで入力層をより少ない層数の隠れ層で表現できるため次元圧縮を可能にしており，データの特徴を抽出することができると考えられている．

事後学習 (Fine-training) では，Pre-training で次元圧縮されたデータの特徴を用いて各クラスに分類を行う．これには Multi Layer Perceptron(MLP) や SVM が使われる．本稿では事前学習に AE，事後学習には MLP を用いて識別を行った．

また，DL のライブラリとして pylearn2[*2] を用いた．pylearn2 は，モントリオール大学で開発されている DL を実装したライブラリである．また，pylearn2 は数値計算ライブラリ theano がベースになっている．theano は微分計算や GPU の計算コードを出力ができることが優れた特徴である．そのため，GPU に対応したソースコードを出力することが可能であり，学習時に GPU を使用することで学習の効率を向上することができる [*3]．

[*2] pylearn2 の HP：http://deeplearning.net/software/pylearn2/
[*3] 本稿で使用した計算機は，CPU が Intel(R) Xeon(R) CPU E5-

4. 教師データについて

教師データの作成するため，ここで改めて良手と悪手について明確に定義する．本稿では，棋譜における着手点を良手とし，着手不可能な場所を含めそれ以外の座標をすべてを悪手と定義する．これは囲碁において容易に学習のために利用可能なデータが過去の対局の記録（棋譜）であるためである．囲碁における機械学習の既存研究でも，筆者の確認する限りではすべて棋譜を用いている．

教師データでは，棋譜の着手点の 1 箇所を良手とし，それ以外の 80 箇所を悪手とする．棋譜における着手の回数だけ教師データを作成する．このとき，良手のデータ数と悪手のデータ数に偏りがあるため，悪手は 80 箇所から無作為に選択した 1 箇所のみを用いる．そのため，1 局面から得られるデータは良手の教師データおよび悪手の教師データをそれぞれ 1 つずつである．

教師データを作成するために，本稿では Computer Go Server(CGOS)[*4] における 2011 年から 2012 年における 9 路盤の対局データ [*5] を用いた．このデータでは比較的強いプログラム同士の対局データで構成されている．訓練データの総数は 656559 局面 (11858 ゲーム) である．9 路盤の棋譜は 19 路盤の棋譜に比べてプロによる対局のデータが少ないため，本稿では強いプログラムによって作られた棋譜を用いることにした．訓練データを用いて，識別機の学習を行う．テストデータとして，訓練データとは別に 83306 局面 (1317 ゲーム) の棋譜を用意した．テストデータを用いて識別機の性能を評価する．

5. 識別機の構成について

本稿では，棋譜を用いて良手と悪手の判別を行うための機械学習を行う．機械学習では入力データの表現方法によって学習結果に大きな違いが生じる．そこで，ここでは識別機の入力データの表現と出力データの表現手法について示す．囲碁では，評価時点での碁盤の状態が入力データとなる．碁盤の表現方法として様々考えられるが，ここでは入力データの表現方法については [3] のデータ表現を参考にすることにした． [3] では，碁盤を黒石の位置，白石の位置を分離して 1 つの碁盤を表現している．本稿もそれに習い碁盤上の黒石がある座標を 1，黒石以外の座標を 0 と表現し，白石の場合も同様に 0，1 で表現した．本稿では，加えて評価する座標 1 点を入力する．評価する座標を 1，それ以外の座標を 0 とする．そのため，一度の識別で 1 座標の評価を行う．出力データの表現は，評価する座標が良手であれば 1，悪手であれば 0 とし 2 値で表現した．

1620 v3 @ 3. 50GHz 16GB Memory, GPU は NVIDIA Tesla C2075(6GB, 448 CUDA cores) である．

*4 CGOS の URL：http://cgos.boardspace.net/

*5 棋譜データの URL：http://www.yss-aya.com/cgos9_201101_201204_2500over.cab

入力データと出力データの表現について (1) で示す．

$$
\begin{aligned}
INPUT &= (B_1, ..., B_{81}, W_1, ..., W_{81}, T1, ..., T_{81}) \\
OUTPUT &= \begin{cases} 1 & \text{(評価対象が良手である場合)} \\ 0 & \text{(otherwise)} \end{cases} \\
B_{\mathrm{i}} &= \begin{cases} 1 & \text{(碁盤の i 番目が黒石である場合)} \\ 0 & \text{(otherwise)} \end{cases} \\
W_{\mathrm{i}} &= \begin{cases} 1 & \text{(碁盤の i 番目が白石である場合)} \\ 0 & \text{(otherwise)} \end{cases} \\
T_{\mathrm{i}} &= \begin{cases} 1 & \text{(碁盤の i 番目が評価の対象である場合)} \\ 0 & \text{(otherwise)} \end{cases}
\end{aligned}
\tag{1}
$$

識別機の入力表現について，図 1 で示す．

深層学習には, 様々な手法が存在する．本稿では, SDA および MLP を用いた深層学習を用いて識別機を構築する．まず，SDA のパラメータについて以下に示す，SDA は事前学習のために用いる．SDA は Denoising-Autoencoder(DAE) を 5 層連ねたものを学習に用いた．それぞれの層について経験的に，162， 81， 41， 21 ，10 ノードに設定した．層の数やノード数については様々考えられるので，本稿の設定よりより良いものが存在することが考えられる．学習時のバッチサイズは 128，学習率は 10^{-4}，学習のエポック数は 100 として学習を行った．次に，MLP のパラメータについて以下で示す．MLP は事後学習のために用いる．つまり，事前学習の結果を用いて実際に分類を行うための学習として MLP を用いた．学習のパラメータとして，学習のバッチサイズは 128，学習率は 0.05 学習のエポック数は 100 として学習を行った．

3 層の NN の構造を図 2，深層学習の識別機の構造を図 3 で示す．

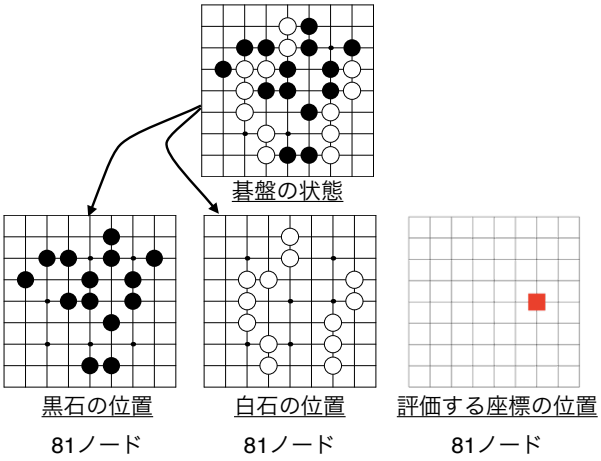

図 1 識別機の入力における碁盤の表現について．

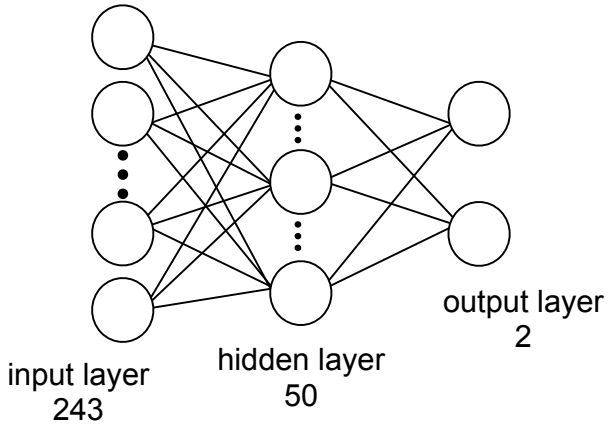

図 2 3 層の NN の構造

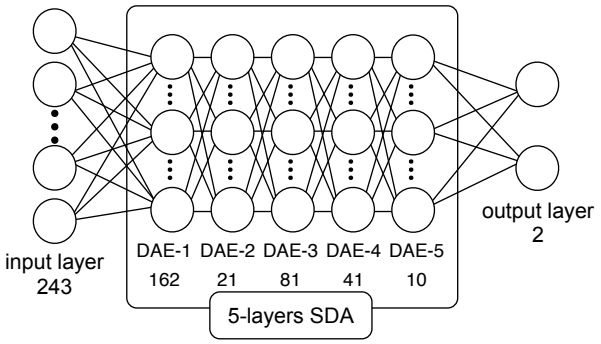

図 3 深層学習の構造

6. 出力の尤度から識別する手法に付いて

本稿で構築した識別機は，出力として良手と悪手の尤度をそれぞれ持つ．その尤度から良手と悪手を判別する手法について次に述べる．

本稿では，以下の 2 つの条件で尤度から良手と悪手を識別することにした．

$$\begin{cases} bad & (L_{bad} > L_{good}) \\ good & (otherwise) \end{cases} \tag{2}$$

$$\begin{cases} bad & (L_{bad} - L_{good} > T) \\ good & (otherwise) \end{cases} \tag{3}$$

ここで L は尤度で L_{good} 良手の尤度，L_{bad} は悪手の尤度である．T は閾値であり，0 から 1 の範囲で任意の値に設定する．まず，(2) はある手の良手と悪手の尤度を比較して，尤度が高いほうに識別する手法である．これは，つまり悪手の尤度がより高ければ悪手，良手の尤度がより高ければ良手になるということである．次に，(3) は悪手の尤度がより大きく，なおかつ尤度の差が T 以上である場合のみ悪手と識別する手法である．この T の取り方は様々考えられるため，T が変化した場合の識別の変化について今後検証する必要がある．本稿では，真の良手を悪手と誤識別しないために十分に大きな値として T を 0.97 に設定した．

表 1 条件 (2) を用いてテストデータでの深層学習 (DL) および 3 層の NN を用いた場合の識別結果.

		True Value	
		Good	Bad
Output of DL	Good	$P_1 = 90.9\%$	$P_2 = 25.7\%$
	Bad	$P_3 = 9.1\%$	$P_4 = 74.3\%$
Output of NN	Good	$P_1 = 87.2\%$	$P_2 = 30.8\%$
	Bad	$P_3 = 12.8\%$	$P_4 = 69.2\%$

表 2 条件 (3) を用いてテストデータでの深層学習 (DL) および 3 層の NN を用いた場合の識別結果.

		True Value	
		Good	Bad
Output of DL	Good	$P_1 = 99.5\%$	$P_2 = 48.3\%$
	Bad	$P_3 = 0.5\%$	$P_4 = 51.7\%$
Output of NN	Good	$P_1 = 99.8\%$	$P_2 = 75.1\%$
	Bad	$P_3 = 0.2\%$	$P_4 = 24.9\%$

7. 識別の結果

学習した識別機について次に示す．学習後の識別機は，1 点の評価のために 0.1 秒程度の処理時間がかかる．このことから，この識別機は時間コストのかかる識別機であることがわかる．また，テストデータにおける識別の様子について表 1 と表 2 で示す．表 1 では (2) の手法を用いた時の結果，表 2 では (3) を用いた時の結果について示す．まず，表 1 の結果を見ると，(2) の条件を用いたときは良手と悪手ともにある程度識別できていることが分かる．特に良手を良手と識別することに優れている．深層学習と 3 層の NN ではわずかに深層学習の性能が良いということがわかる．次に，表 2 の結果を見ると，(3) の条件を用いた時は特に良手を良手と識別することに優れているということがわかる．良手を悪手と誤識別することが少ない識別機となっていることがわかる．また，深層学習では NN に比べて悪手を悪手と識別することについて優位にあることが分かる．また，(2) の枝刈りの様子について図 4 ，(3) の枝刈りの様子について図 5 で示す．

8. 識別機を枝狩りに使用した場合の様子について

本稿の識別機を用いて，単純なモンテカルロ木探索で枝刈りを行った．ここで，対戦したプログラムについて述べる．対局プログラムのベースとなるプログラムはモンテカルロ木探索を用いた．モンテカルロ木探索はノード選択のためのアルゴリズムとして，UCB1 を用いる UCT アルゴ

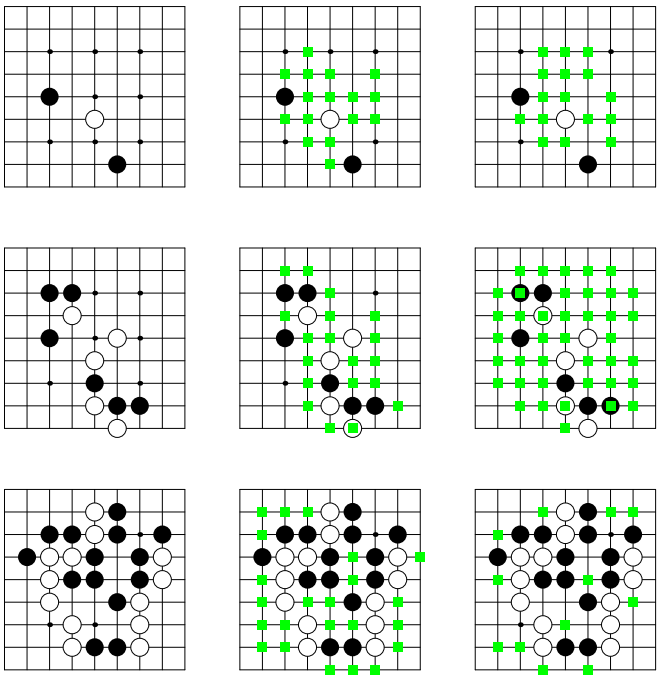

図 4 条件 (2) を用いて良手と悪手を識別する様子．緑の四角で示す点が良手と判別された箇所．それ以外はすべて悪手と判別した箇所．左から評価する局面，深層学習の評価，3 層の NN の評価．

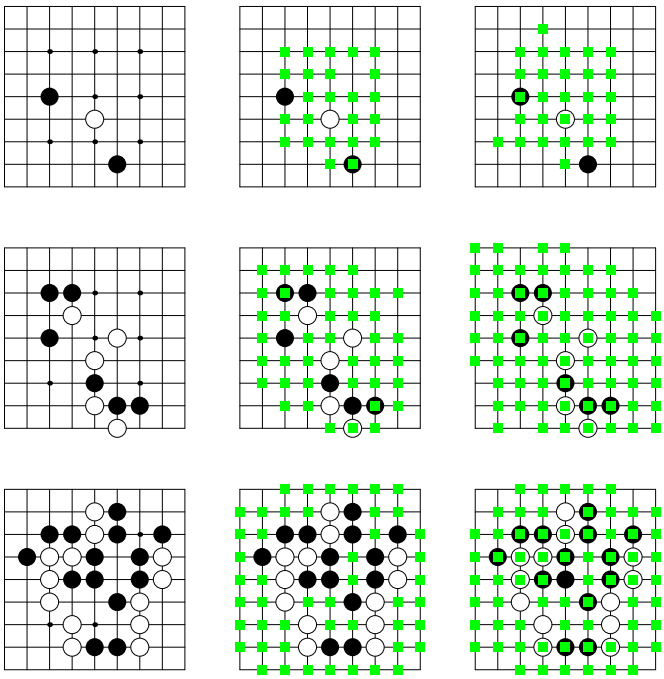

図 5 条件 (3) を用いて良手と悪手を識別する様子．緑の四角で示す点が良手と判別された箇所．それ以外はすべて悪手と判別した箇所．左から評価する局面，深層学習の評価，3 層の NN の評価．

リズム [1] で実現した．プレイアウトは完全な乱数で実現したプログラムを用いた．木探索では，深さの制限を 10 として 1 手にかけるプレイアウトの回数は 10,000 回とした．

表 3 実際に囲碁プログラムを用いた時の枝刈りの効果について．

	Wining rate	
	Black	White
Plane-mc	66%	34%
Pruning-mc	74%	42%

枝刈りなしのプログラム (Plane-mc) と枝刈りありのプログラム (Pruning-mc) を用意して，それぞれを対局させた．また，比較のために枝刈りなしのプログラム同士の対局も行った．その結果をまとめた結果を表 3 で示す．これを見ると，枝刈りを実施した場合の勝率が，黒石と白石でそれぞれ 8%向上していることが分かる．

また，枝刈り時のプログラムの対局を 2 局を例として図 6 と図 7 で示す．

9. まとめ

本稿では，良手と悪手を識別するために SDA を用いた深層学習と 3 層の NN による学習で識別機を構築した．どちらも悪手に関しては非常に高い信頼性で得ることができることを確認した．また，悪手の識別をモンテカルロ木探索の枝刈りで使用したときの効果について検証した．枝刈りが無い状態のプログラムと枝刈りを行ったプログラムで実際にゲームを行ったところ，黒石でも白石でも勝率が向上することが確認できた．

今後の課題として，この識別機を用いてさらに強い囲碁プログラムに応用した場合の効果について検証することが挙げられる．その際に，単純な枝刈りではなく前向きな枝刈り手法 Progressive Wedding を用いることが考えられる．また，[3] や [4] では CNN を用いた識別機では棋譜の手を高い割合で再現することに成功しているため，学習の方法や学習機そのものを変更してより性能の高い識別機を構築することが考えられる．

参考文献

[1] B. Bernd: Monte Carlo Go. Technical report, Department of Physics, Syracuse University, 1993.

[2] N. Araki: Monte Carlo simulation adjusting, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[3] C. Clark and A. Storkey: Teaching deep convolutional neural networks to play Go. Cornell University Library, arXiv:1412.3409v1, 2014. Available from http://arxiv.org/pdf/1412.3409v1.pdf.

[4] A. Huang , I. Sutskever and D. Silver: Move evaluation in Go using deep convolutional neural networks. Cornell Univer-sity Library, arXiv:1412.6564v2, 2015. Available from http://arxiv.org/pdf/1412.6564v2.pdf.

[5] S. Sato, O. Yamamoto: On elimination of bad moves for monte carlo Go using a neural network, *ICT International Student Project Conference*, 3C-4, 2015.

[6] G. E. Hinton and R. R. Salakhutdinov: Reducing the dimensionality of data with neural networks. Science 313 (5786) 504-507, 2006.

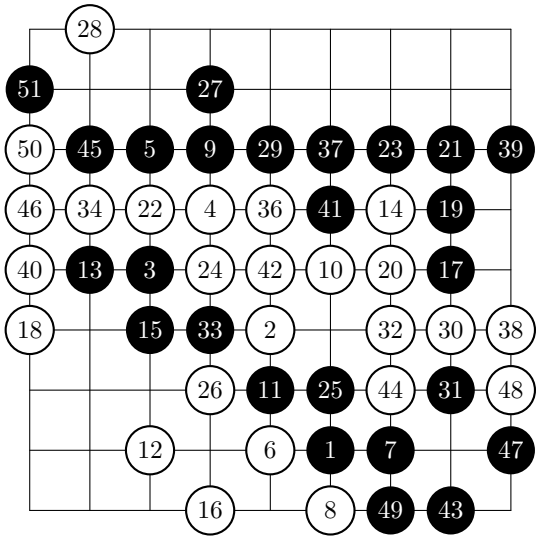

図 6 枝刈りありのプログラム (Pruning-mc) と枝刈りなしのプログラム (Plane-mc) の対局例 1．51 手まで．Pruning-mc は白石，Plane-mc は白石.

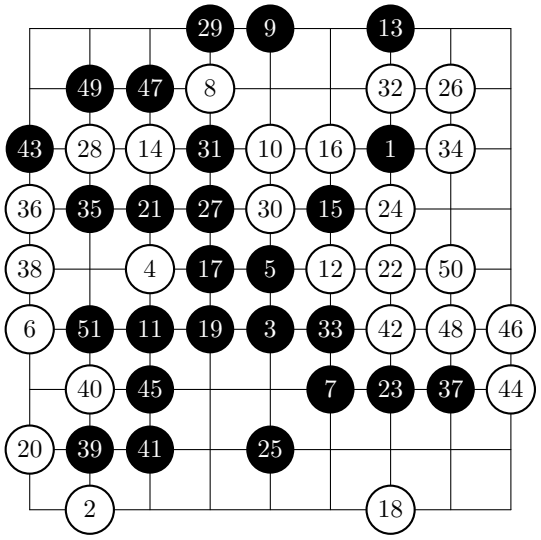

図 7 枝刈りありのプログラム (Pruning-mc) と枝刈りなしのプログラム (Plane-mc) の対局例 2．51 手まで．Pruning-mc は黒石，Plane-mc は黒石.

[7] P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol: Extracting and Composing Robust Features with Denoising Autoencoders. *Proceedings of the Twenty-fifth International Conference on Machine Learning* (ICML ‘ 08)，pages 1096 - 1103, ACM, 2008.

[8] LISA lab: Multilayer Perceptron - DeepLearning 0.1 documentation. http://deeplearning.net/tutorial/mlp.html.