第 34 回世界コンピュータ将棋選手権
dlshogi with HEROZ アピール文章

山岡忠夫
加納邦彦
大森悠平
2024/3/26

※下線部分は、第 33 回世界コンピュータ将棋選手権からの差分を示す。

1　dlshogi のアピールポイント

dlshogi は、ディープラーニングを使用した将棋 AI である。

2017 年より、AlphaGo の手法を参考に開発を行っている。

ディープラーニング系の将棋 AI は、大局観に優れており、序中盤の形勢判断が従来型将棋 AI と比べて正確であるという特徴がある。一方、終盤の読みが重要になる局面では、従来型将棋 AI の方が正確な場合がある。

dlshogi は、終盤の課題に対処するために、独自の工夫を行っている。

具体的には、「MCTS の葉ノードでの短手数の詰み探索」、「ルート局面で df-pn による長手数の詰み探索」、「勝敗が確定したノードのゲーム木への伝播」、「PV 上の局面に対する長手数の詰み探索」、「強化学習時に初期局面集を使用して局面の多様性を確保する」、「強化学習時に df-pn により詰み探索を行い詰みを報酬とする」という工夫を行っている。

これらのいくつかは、現在、dlshogi 以外のディープラーニング系の将棋 AI には取り入れられているが、dlshogi 以前にこれらを導入しているディープラーニング系の将棋 AI はなかった。

<u>今大会に向けては、新しい手法で定跡の自動生成を行った。</u>

2　チーム参加について

今大会では、HEROZ チームとして参加する。

3　dlshogi の特徴

- ディープラーニングを使用
- 指し手を予測する Policy Network
- 局面の勝率を予測する Value Network
- 入力特徴にドメイン知識を積極的に活用
- モンテカルロ木探索
- 未探索のノードの価値に親ノードの価値を使用
- GPU によるバッチ処理に適した並列化

- 自己対局による強化学習
- 詰み探索結果を報酬とした強化学習
- 既存プログラムを加えたリーグ戦による強化学習
- 既存将棋プログラムの自己対局データを混ぜて学習
- 既存将棋プログラムの自己対局データを使った事前学習
- ブートストラップ法による Value Network の学習
- 引き分けも含めた学習
- 指し手の確率分布を学習
- 同一局面を平均化して学習
- 評価値の補正
- SWA(Stochastic Weight Averaging)
- 末端ノードでの短手順の詰み探索
- ルートノードでの df-pn による長手順の詰み探索
- 勝敗が確定したノードのゲーム木への確実な伝播
- PV 上の局面に対する長手数の詰み探索
- 序盤局面の事前探索（定跡化）
- 定跡作成時に floodgate の棋譜の統計を利用した確率分布を方策に利用
- マルチ GPU 対応（NVIDIA A100×8 を使用予定）
- TensorRT を使用
- Optuna による探索パラメータの最適化
- 確率的な Ponder
- ノードのガベージコレクションとノード再利用処理の改良
- 飛車と角の利きのビット演算
- 2 値の入力特徴量を 1 ビットで転送することで推論のスループットを向上
- Stochastic Multi Ponder

4　使用ライブラリ

- Apery[1]（WCSC28）
  →局面管理、合法手生成のために使用

4.1　ライブラリの選定理由

本プログラムは、将棋におけるディープラーニングの適用を検証することを目的としており、学習局面生成、局面管理、合法手生成については、使用可能なオープンソースがあれば使用する方針である。そのため、学習局面を圧縮形式(hcpe)で生成する機能を備えていて、合法手生成を高速に行える Apery を選定した。

[1] https://github.com/HiraokaTakuya/apery

5　各特長の具体的な詳細（独自性のアピール）

5.1　ディープラーニングを使用

DNN(Deep Neural Network)と MCTS を使用して指し手を生成する。

従来の探索アルゴリズム(α β法)、評価関数(3 駒関係)は使用していない。

5.2　Policy Network

局面の遷移確率を Policy Network を使用して計算する。

Policy Network の構成には、Residual Network を使用した。

入力の畳み込み 1 層と、ResNet 30 ブロック(畳み込み 2 層で構成)と出力層の合計 62 の畳み込み層で構成した。フィルターサイズは 3（入力層の持ち駒のチャンネルのみ 1)、フィルター数は 384 とした。

5.3　Value Network

局面の勝率を Value Network を使用して計算する。

Value Network は、Policy Network と出力層以外同じ構成で、出力層に全結合層をつなげ、シグモイド関数で勝率を出力する。

5.4　入力特徴にドメイン知識を積極的に活用

Alpha Zero では、入力特徴に呼吸点のような囲碁の知識を用いずに盤面の石の配置と履歴局面のみを入力特徴とすることで、ドメイン知識なしでも人間を上回ることが示された。しかし、その代償として、入力特徴にドメイン知識を活用した AlphaGo Lee/Master に比べて倍のネットワークの層数が必要になっている。AlphaGo Zero の論文の Figure 3 によると、ネットワーク層数が同一のバージョンでは Master を上回る前にレーティングが飽和している。

強い将棋ソフトを作るという目的であれば、積極的にドメイン知識を活用した方が計算リソースを省力化できると考えられる。

そのため、本ソフトでは、入力特徴に盤面の駒の配置の他に、利き数と王手がかかっているかという情報を加えている。それらの特徴量が学習時間を短縮する上で、有効であることは実験によって確かめている。

5.5　モンテカルロ木探索

対局時の指し手生成には、Policy Network と Value Network を活用したモンテカルロ木探索を使用する。

ノードを選択する方策に、Policy Network による遷移確率をボーナス項に使用した PUCT アルゴリズムを使用する。PUCT アルゴリズムは、AlphaZero の論文[2]の疑似コードに記述さ

[2] http://science.sciencemag.org/content/362/6419/1140

れた式を使用した。

また、末端ノードでの価値の評価に、Value Network で計算した勝率を使用する。

通常のモンテカルロ木探索では、末端ノードから複数回終局までプレイアウトを行った結果（勝率）を報酬とするが、将棋でランダムなプレイアウトは有効ではないため、プレイアウトを行わず Value Network の値を使用する。

5.6　未探索のノードの価値に親ノードの価値を使用

モンテカルロ木探索の UCB の計算時に、未探索の子ノードがある場合、そのノードの価値に何らかの初期値を与える必要がある。子ノードの価値は親ノードの価値に近いだろうという将棋のドメイン知識を利用し、それまでの探索で見積もった親のノードの価値を動的に初期値として使用する。ただし、ノードの訪問回数が増えるに従い、その価値の減衰を行い、幅より深さを優先した探索を行う(FPU reduction)。

5.7　GPU によるバッチ処理に適した並列化

複数回のシミュレーションを順番に実行した後、それぞれのシミュレーションの末端ノードの評価をまとめて GPU でバッチ処理する。その後、評価結果をそれぞれのシミュレーションが辿ったノードにバックアップする。以上を一つのスレッドで行うことで、マルチスレッドによる実装で課題となる GPU の計算後にスレッドが再開する際にリソース競合が起きる問題（大群の問題）を回避する。

GPU で計算中は、CPU が空くため、同じ処理を行うスレッドをもう一つ並列で実行する。2 つのスレッドが相互に CPU と GPU を利用するため、利用効率が高い処理が可能となる。

5.8　自己対局による強化学習

事前学習を行ったモデルから開始して、AlphaZero[3]と同様の方式で強化学習を行う。自己対局により教師局面を生成し、その教師局面を学習したモデルで、再び教師局面を生成するというサイクルを繰り返すことでモデルを成長させる。

2018 年の大会で使用した elmo で生成した教師局面で収束するまで学習したモデルに比べて、自己対局による強化学習によって有意に強くすることができた。

5.9　詰み探索結果を報酬とした強化学習

自己対局時に終局まで対局を行うと、モンテカルロ木探索の特性上、詰むまでの手順が長くなる傾向がある。勝率予測に一定の閾値を設けることで、終局する前に勝敗を判定することで対局時間を短縮できるが、モデルの精度が低いうちは誤差が大きいため、学習精度に影響する。

この課題の対策として、df-pn による高速な長手数の詰み探索の結果を報酬とした。単純に

[3] https://arxiv.org/abs/1712.01815

すべての局面で詰み探索を行うと、自己対局の実行速度が大幅に落ちてしまう。自己対局は複数エージェントに並列で対局を行わせ、各エージェントからの詰み探索の要求をキューに溜めて、詰み探索専用スレッドで処理するようにした。エージェントが GPU の計算待ちの間に詰み探索が完了する。エージェントが探索している局面は別々のため、時間のかかる詰み探索の要求が集中することは少ない。これにより自己対局の速度を大幅に落とすことなく長手数の詰み探索を行えるようになった。

5.10 既存プログラムを加えたリーグ戦による強化学習

自分自身のプログラムのみで強化学習を行うと戦略に弱点が生まれる可能性がある。弱点をふさぐには多様なプログラムによるリーグ戦が有効だが、複数のエージェントを学習するにはエージェント数の分だけ余分に計算資源が必要になる。

計算資源を省力化して、リーグ戦の効果を得るために、オープンソースで公開されている既存の将棋プログラムを 1/8 の割合でリーグに加えて強化学習を行うようにした。

5.11 既存将棋プログラムの自己対局データを使った事前学習

本プログラムを使用して、Alpha Zero と同様に、ランダムに初期化されたモデルから強化学習を行うことも可能だが、使用可能なマシンリソースが足りないため、スクラッチからの学習は行わず、既存将棋プログラムの自己対局データを教師データとして、教師あり学習でモデルの事前学習を行う。

教師データには、elmo で生成した自己対局データを使用した。

5.12 既存将棋プログラムの自己対局データを混ぜて学習

以前の dlshogi は、入玉宣言勝ちできる局面でなかなか入玉宣言勝ちを目指さないという課題があった。

自己対局では入玉宣言勝ちの棋譜が少ないため、それを補うため既存将棋プログラム(水匠)の自己対局で、入玉宣言勝ちの棋譜を生成し、dlshogi の自己対局のデータに混ぜて学習した。

5.13 ブートストラップ法による Value Network の学習

Value Network の学習の損失関数は、勝敗を教師データとした交差エントロピーと、探索結果の評価値を教師データとした交差エントロピーの和とした。

このように、本来の報酬（勝敗）とは別の推定量（探索結果の評価値）を用いてパラメータを更新する手法をブートストラップという。

経験的にブートストラップ手法は、非ブートストラップ手法より性能が良いことが知られている。

5.14 引き分けも含めた学習

将棋はルールに引き分けがあるゲームであるため、引き分けも正しく学習できる方が望ましい。そのため、自己対局で引き分けとなった対局も学習データに含めて学習した。

5.15 指し手の確率分布を学習

以前の dlshogi では、指し手のみを学習していたが、AlphaZero と同様に、自己対局時で MCTS で探索した際のルート局面の子ノードの訪問回数に従った確率分布を学習するように変更した。確率分布を学習することで、最善手と次善手の行動価値が近い場合に、次善手の行動価値を正しく学習できるようになる。

確率分布を学習することで、floodgate の棋譜に対する一致率が向上することが確認できたが、対局して強さを計測すると弱くなるという現象が確認できた。原因は、モデルの方策の出力の性質が変わるため、探索パラメータの調整が必要なためであった。Optuna を使用して探索パラメータを最適化(5.27 参照)することで、指し手のみを学習したモデルよりも強くすることができた。

5.16 同一局面を平均化して学習

自己対局では、序盤の同一の局面の教師データが多く生成される。それらの重複した局面を別のサンプルとして学習すると、モデルの学習に偏りが起きる。

局面の偏りをなくするために、同一の局面を集約し、指し手の確率分布と勝敗を平均化し、1 サンプルとして学習した。

5.17 評価値の補正

自己対局で生成するデータには、MCTS で探索して得られた勝率(最善手の価値)を局面の評価値を記録し、学習時にブートストラップ項(5.13 参照)として使用している。記録した評価値(勝率)が、実際の対局の結果から算出した勝率と一致しているか調べたところ、乖離しているという現象が確認できた。そのため、評価値を実際の自己対局での勝率に合うように、補正を行った。

5.18 SWA(Stochastic Weight Averaging)

画像認識の分野でエラー率の低減が報告されている手法である、SWA(Stochastic Weight Averaging)をニューラルネットワークの学習に取り入れた。一般的なアンサンブル手法では、推論結果の結果を平均化するが、SWA では学習時に一定間隔で重みを平均化することでアンサンブルの効果を実現する。

5.19 末端ノードでの短手順の詰み探索

モンテカルロ木探索の末端ノードで、5 手の詰み探索を行い、詰みの局面を正しく評価できるようする。並列化の方式により、GPU で計算中の CPU が空いた時間に詰み探索を行う

ため、探索速度が落ちることはない。

5.20 ルートノードでの df-pn による長手数の詰み探索

モンテカルロ木探索は最善手よりも安全な手を選ぶ傾向があるため詰みのある局面で駒得になるような手を選ぶことがある。

対策として、詰み探索を専用スレッドで行い、詰みが見つかった場合はその手を指すようにする。

詰み探索は、df-pn アルゴリズムを使って実装した。優越関係、証明駒、反証駒、先端ノードでの 3 手詰めルーチンにより高速化を行っている。

5.21 勝敗が確定したノードのゲーム木への確実な伝播

モンテカルロ木探索で構築したゲーム木の末端ノードで詰みが見つかった場合、その結果をゲーム木に伝播して利用する。つまり、モンテカルロ木探索に、AND/OR 木の探索を組み合わせて、詰みの結果を確実にゲーム木に伝播するようにする。

5.22 PV 上の局面に対する長手数の詰み探索

ディープラーニング系の将棋 AI は、選択的に探索を行うために、終盤の局面で読み抜けがあると、頓死することある。

頓死を防ぐため、PV 上の局面に対して、df-pn による長手数の詰み探索を行い、詰みが見つかった場合、局面の価値を更新するようにする。

5.23 序盤局面の事前探索（定跡化）

出現頻度の高い序盤局面は、対局時に探索しなくても、事前に探索を行い定跡化しておくことができる。また、事前に探索することで、対局時よりも探索に時間をかけることができる。

ゲーム木は指数関数的に広がるため、固定の手数までの定跡を作成するよりも、有望な手順を選択的に定跡に追加する方が良い。自分が指す手は、1 つ局面につき最善手を 1 手（または数手）登録し、それに対する応手は、公開されている定跡や棋譜の統計情報を使って確率的に選択する。その手に対して、また最善手を 1 手（または数手）登録する。この手順により、頻度の高い局面については深い手順まで、頻度の低い局面については短い手順の定跡を作成することができる。

5.24 定跡作成時に floodgate の棋譜の統計を利用した確率分布を方策に利用

定跡を自分自身の探索のみで作成した場合、読み抜けがあった場合に定跡を抜けた後に不利な局面になる恐れがある。そのため、モンテカルロ木探索の PUCT の計算で、方策ネットワークの確率分布と floodgate の棋譜の統計を利用した確率分布を平均化した確率分布を利

用し、致命的な読み抜けを防止する。

5.25 マルチ GPU 対応

複数枚の GPU を使いニューラルネットワークの推論を分散処理する。

「5.7 GPU によるバッチ処理に適した並列化」の方式により、GPU ごとに 2 つの探索スレッドを割り当てることで、GPU を増やすことでスケールアウトすることができる。ノードの情報は、すべてのスレッドで共有する。

確認できている範囲で 4GPU までほぼ線形で探索速度を上げることができている。

5.26 TensorRT を使用

モデルの学習にはディープラーニングフレームワークとして PyTorch を使用しているが、対局プログラムには、推論用ライブラリの TensorRT を使用する。

TensorRT を使うことで、事前にレイヤー融合などのニューラルネットワークの最適化を行うことで、推論を高速化することができる。TensorCore に最適化されており、TensorCore を搭載した GPU では CUDA+cuDNN で推論を行う場合より、約 1.33 倍の高速化が可能になる[4]。

また、対局の実行環境にディープラーニングフレームワークの環境構築を不要とすることを目的とする。

5.27 Optuna による探索パラメータの最適化

PFN により公開された Optuna[5]を使用して、モンテカルロ木探索の探索パラメータ（PUCT の定数、方策の温度パラメータ）を最適化した。

Optuna は、主にニューラネットワークの学習のハイパーパラメータを最適化する目的で利用されるが、将棋エンジン同士の連続対局の勝率を目的関数として、探索パラメータの最適化に使えるようにするスクリプト[6]を開発した。Optuna の枝刈り機能により、少ない対局数で収束させることができる。

5.28 確率的な Ponder

モンテカルロ木探索は確率的にゲーム木を成長させる。その特性を活かして、相手が思考中に、相手局面からモンテカルロ木探索を行うことで、確率的に相手の手を予測して探索を行うことができる。予測手 1 手のみを Ponder の対象とするよりも、効率のよい Ponder が実現できる。

---

[4] https://tadaoyamaoka.hatenablog.com/entry/2020/04/19/120726

[5] https://optuna.org/

[6] https://github.com/TadaoYamaoka/DeepLearningShogi/blob/master/utils/mcts_params_optimizer.py

5.29 ノードのガベージコレクションとノード再利用処理の改良

世界コンピュータ将棋オンライン大会でノード再利用に 10 秒以上かかる場合があるこがわかったため、ノード再利用の方式の見直しを行った。

以前は、オープンアドレス法でハッシュ管理を行っており、ルートノードから辿ることができないノードをすべてのハッシュエントリに対して線形探索してノードの削除をおこなっていた。

これを、Leela Chess Zero のゲーム木の管理方法[7]を参考に、ゲーム木をツリーで管理を行うようにし、ルートの兄弟ノードをガベージコレクションする方式に変更した。ノードの合流の処理が行えなくなるという欠点があるが、ノード再利用を短い時間で行えるようになった。

5.30 飛車と角の利きのビット演算

第 31 回世界コンピュータ将棋選手権の Qugiy のアピール文章[8]による、飛車、角の利きをビット演算により求める方法を実装した（実装はやねうら王のソースコードを参考にした)。ZEN2 の CPU で NPS が約 1% 向上した。

5.31 2 値の入力特徴量を 1 ビットで転送することで推論のスループットを向上

マルチ GPU を使用した場合、4GPU 以上では CPU と GPU 間の帯域がボトルネックになるため、2 値の入力特徴量を float の代わりに、1bit で表現し、GPU にビットで転送後、GPU 側で CUDA のプログラムでバッチ単位に並列に float に戻す処理を実装した。こうすることで、転送量が削減でき、NPS が 36.6%向上した。

5.32 Stochastic Multi Ponder

相手番に、相手番の局面から探索を行う Stochastic Ponder（5.28 参照）と、次の相手の指し手を複数予測し、並列に分散して探索を行う Multi Ponder を組み合わせて使う。

shotgun で実装されていた Multi Ponder[9]では、技巧 2 の Multi PV の結果を利用しているが、dlshogi の Stochastic Ponder では、ほとんどの場合、相手局面でのゲーム木が展開済みであり、ルートの子ノードの訪問回数を参照することで、有望な予測手を N 手取得することができる（ゲーム木が未展開の場合は、方策ネットワークの推論結果を使用する)。

また、予測した N 手以外の手が指された場合、Stochastic Ponder でも並列に探索を行っているため、Multi Ponder を使用しない場合と遜色のない手を指すことができる。

ponderhit した場合、次の局面の指し手予測の第一候補をその ponderhit したエンジンに

[7] https://tadaoyamaoka.hatenablog.com/entry/2020/05/05/181849
[8] https://www.apply.computer-shogi.org/wcsc31/appeal/Qugiy/appeal_210518.pdf
[9] http://id.nii.ac.jp/1001/00199872/

割り当てることで、前回の探索結果を再利用する。

やねこま王チーム PR 文章

ペタショック定跡

以下のブログ記事に書いたように 3000 万局面程度生成した。

将棋 AI の 2024 年は大規模定跡時代
https://yaneuraou.yaneu.com/2024/01/14/the-era-of-large-scale-book-in-shogi-ai/

DL 系の将棋 AI で読み抜けする件の改良

今回、探索部には DL(Deep Learning)系の将棋 AI であるふかうら王(自作の dlshogi 互換エンジン)を用いる。

ふかうら王は AlphaZero 型の将棋 AI である。

AlphaZero 型の DL 系の将棋 AI ではしばしば読み抜けするという問題がある。

これは Policy Network(方策予測のためのニューラルネットワーク)が指し手の実現確率を出力するのだが、好手に対して極めて 0 に近い確率を出力することがあるからだと思う。

このような場合、その指し手の先の変化を全く読まないので読み抜けする。0 に近い確率が出力された場合であっても少しぐらい読むべきだと思うが、そうしてしまうと読まなければならない枝が増えすぎてしまい、Policy Network を用意している意味がなくなってしまう。

そこで特定の条件に合致した場合のみ、0 に近い確率が出力された場合であってもその先の変化を読むようにしようかと思っている。

WCSC34 W@ndre7 アピール文書(仮)

Dated 2024/02/07

気が付くと今回で世界コンピュータ将棋選手権に7回目の出場だそうです。あっという間ですね。
これまでのアピール文等を振り返ってみると、ディープ系に取り組むといいながら、なかなか実践していないなというのを再認識しました。
そこで、今年はdlshogi系をベースとし、いろいろ触りながら改造していく予定です。今はResNetと戯れている最中。

過去のアピール文

WCSC33 W@nderER 1 2 3(pdf)
WCSC32 W@nderER 1 2 3(pdf)
WCSC31 W@nderER 1 2(pdf) 2(docx)
WCSC30 W@nderER 1 2 3 (※WCSC30は中止で代わりにWCSOC2020が開催)
WCSC29 W@ndre 1 2 3 4
WCSC28 W@ndre 1 2
SDT5 W@ndre SDT5(pdf)

二番絞り

二番絞りの誕生経緯については2022年のアピール文などを参考に頂ければ幸いであるが基本的には最高精度を目指す大きなモデルを作成しようとする試みである.

2022年度は幸いにも準優勝という好結果を得た．準備段階では手ごたえがなかったがその後の計測で大変高精度なものが完成していたことが分かった．今年度は新しい試みを加えたがまともな計測に至っていない。もし旧バージョンより弱いと判断した場合は旧バージョンで出場する可能性がある.

ちなみに，2022度版の局面評価精度は驚愕の域に達しており，既発表であるが一手の局面展開も行わず将棋倶楽部24でレート2949，八段認定頂いている．前人未到の領域といって過言ではないだろう.

また，電竜戦ハードウェア統一戦においても準優勝となり，記念に2017年に行われたハードウェア統一戦の第5回電王トーナメントを思い起こしGTX1080Tiの二番絞りをfloodgateに投入したところ短期レート4500台を記録した．（もちろん一時的なものでありしばらくしてレートは4100〜4200程度で落ち着いた）

今年度はこれを上回ることを目指しているが上記の通り昨年に続き未計測である.

参考：

芝，「将棋のPV-MCTSに向けた深層学習モデルの最適化」，第45回ゲーム情報学研究会

芝，「探索アルゴリズムに適した時間利用に関する研究」，第46回ゲーム情報学研究会

第32回世界コンピュータ将棋選手権，https://bleu48.hatenablog.com/entry/2022/05/06/145915

芝，「コンピュータ将棋における高精度な深層学習モデル」，ゲームプログラミングワークショップ2022

二番絞り＠将棋倶楽部24の戦型分析，https://bleu48.hatenablog.com/entry/2023/03/08/062634

二番絞りの計測の件（GX1080Ti編），https://bleu48.hatenablog.com/entry/2023/03/09/132936

第34回世界コンピュータ将棋選手権 「東横将棋」アピール文書

2024.3.31

東横コンピュータ将棋部

定跡とNNUE評価関数の極北を目指しています。

・従来の強化学習手法に加え独自にNNUE評価関数を強化。今回からマメットブンブク評価関数（halfkp_1024x2-8-32）を使用しています。

・手作業による定跡の生成。角換わり定跡と相掛かり定跡に主眼を置いています。

・探索部はやねうら王、いわゆるやねうら王チルドレンです。最新のやねうら王の使用を予定しています。

よろしくお願いいたします。

ノミというのがおりますなｗｗｗちっぽけな虫ケラのノミですなｗｗｗ

あの虫は我々巨大で頭のいい人間にところかまわず攻撃を仕掛けて戦いを挑んでくるんですなｗｗｗ

巨大な敵に立ち向かうノミｗｗｗこれは「勇気」と呼べるんですかなｗｗｗ

ノミどものは「勇気」とは呼べませんなｗｗｗ

では「勇気」とはいったい何なんですかなｗｗｗ

「勇気」とは「怖さ」を知ることｗｗｗ「恐怖」を我が物とすることなんですなｗｗｗ

人間讃歌は「勇気」の讃歌ｗｗｗ人間のすばらしさは勇気のすばらしさｗｗｗ

いくら強くてもこいつら屍生人は「勇気」を知らんのですなｗｗｗ

ノミと同類ですなｗｗｗ

んんんｗｗｗ

今年も性懲りもなく出るんですなｗｗｗ

・役割論理とは

第33回世界コンピュータ将棋選手権で新人賞を受賞する快挙()で完全かつ最終的に確立された論理ですなｗｗｗ

しょぼい定跡とマメットブンブク評価関数の強化と元高級スリッパ（粗大ゴミ）の微妙な火力によって評価値ダメージレースを制する必勝の戦術ですなｗｗｗ

クラウド重課金(笑)はもうやめましたぞｗｗｗ
おうちパソコン3990Xでどこまでやれるか見ものですなｗｗｗ

・定跡について
floodgateその他で収集した棋譜やら何やら適当にぐちゃまぜにして脳を焼きながら手作業で調整修正した居飛車定跡「1.21ジゴワット火葬砲定跡」を使用しますぞｗｗｗ

・戦型について
角換わり、相掛かり共に先手必勝が確定しましたなｗｗｗ後手番は対策が急務となっておりますぞｗｗｗ

ぶっちゃけ対策なんかできませんなｗｗｗ
しっかり定跡をキメてこられるとまったく太刀打ちできませんぞｗｗｗありえないｗｗｗ
特に角換わりは絶対に拒否した方がいいですなｗｗｗ

ちなみに振り飛車は先手後手ともに必敗ですなｗｗｗ埴輪定跡は徹底的に対策するしかありえないｗｗｗ

横歩取り：先手必勝ですなｗｗｗ
一手損角換わり：先手必勝ですなｗｗｗ
雁木：先手番でわざわざ指す必要はなさそうですなｗｗｗえぐいですなｗｗｗ
矢倉：先手番でわざわざ指す必要はなさそうですなｗｗｗ矢倉は終わりましたｗｗｗ
相振り飛車：なんで相手がありがたくも不利飛車を指してくれているのにこちらも振ってやる必要があるんですかなｗｗｗありえないｗｗｗ
筋違い角：後手の振り飛車党への嫌がらせの精神攻撃ですなｗｗｗそれ以上でもそれ以下でもありませんぞｗｗｗだいたい負けますなｗｗｗ

まずは後手番でどこまでダメージを最小限にするかが重要ですなｗｗｗ
現状ほぼ先手後手が同じ割合になるらしいので後手番でも勝てそうな相手(酷い)に必然力で対戦すること

が重要ですぞｗｗｗ

互角の分かれで逃げられればNNUE型評価関数の終盤の強さと元高級スリッパの火力で踏み潰すだけですなｗｗｗ

もちろん定跡を整備しなければdl系やや○こま王()や水○匠などの強豪には手も足も出ませんぞｗｗｗ

・必然力とは

論者を圧倒的勝利に押し上げる力ですなｗｗｗ

強豪に 絶 対 に 先手を引く、後手番での当たりが良いｗｗｗ裏街道最高ｗｗｗなどは必然力とされていますなｗｗｗ

ヤーティ神への信仰によって得られる加護とされていますぞｗｗｗ

やはりこれが最も重要なファクターとなっていますなｗｗｗ

結局「評価関数の強化」と「定跡の強化」という極めて当たり前の結論に至る訳ですなｗｗｗ

将来的にはdl系とマメットブンブク形式（halfkp_1024x2-8-32）のハイブリッド＋強力な定跡が主流になっていくのではないですかなｗｗｗ知らんけどｗｗｗ

現時点では定跡強化が勝利の鍵になりそうですなｗｗｗ

・評価関数について

ついに待望のマメットブンブク評価関数（halfkp_1024x2-8-32）を使用しますぞｗｗｗPytorchｗｗｗ

もうどこもhalfkp_1024x2-8-32ばかりで特にアドはありませんなｗｗｗ入玉どうするんですかなｗｗｗ

知らんがなｗｗｗ

なお標準NNUE評価関数はもう完全にサチっててオワコンでもはや観る将が藤井聡太の将棋を観戦する時にしか使い道がありませんなｗｗｗ

標準NNUEで粘る場合でも水匠5はもちろんHaoやBLOSSOMより強くないとお話になりませんなｗｗｗゴミｗｗｗ

ﾅﾝﾄｶ改？知りませんなｗｗｗ

もちろん振り飛車評価関数は総合的にロジックするまでもなくありえないｗｗｗ

・使用予定の評価関数

Grampus_HD2_20220228：とんいる人民共和国()でこの世の誰も体験したことのない最も厳しい無慈悲な鉄槌を受けたマメットブンブク評価関数ですなｗｗｗ

誰の評価関数なんですかなｗｗｗ

もう評価関数に手を付けている暇などなさそうなので苦渋の選択ですぞｗｗｗ

やっぱりちょっとだけ手を入れる予定ですぞｗｗｗ

・シードについて

今回は第6シードですなｗｗｗ感謝しかありえないｗｗｗ

全体の実力が底上げされているので一次予選から修羅場になる可能性も十分にありますなｗｗｗそして1枠は某7995WX枠が確定しておりますぞｗｗｗ

・東横将棋について

A.東横将棋はGrampusですか？

Q.違いますなｗｗｗ東横将棋は東横将棋であってそれ以上でもそれ以下でもありませんぞｗｗｗ

第３４回世界コンピュータ将棋選手権
「あすとら将棋」アピール文書

令和６年４月
恒岡 正年

1. 全体の構成
dlshogi の探索部、学習部を改造して利用。独自構造の model を採用。

2. 独自に実装した部分
Network 構造、学習方法、探索部の改造、model の生成の工夫等。

3. 開発動機
深層学習の勉強を兼ねて dl 系将棋 AI の model の学習を始めた。
WCSC33 に参加して決勝戦で全敗。上位ソフトと良い勝負をしたい。

4. 主な開発内容
独自構造の model の作成・学習方法
棋譜生成：5kpo～20kpo 程度の物を中心に現在約 55 万個の棋譜作成
学習には、上記生成した棋譜以外に GCT の学習用データを利用
探索パラメータの調整(optuna 利用)

５. 定跡生成
前回は手入力の定跡だったが、今回は条件を満たす棋譜を集め定跡を生成した。
条件付きで選別した自己生成棋譜及び Floodgate の棋譜から作成した。

千日手の棋譜を含めるべきかどうかを検討した結果、Draw_Value_Black/White を調整すれば千日手の棋譜を含めなくても良いとの結論になった。
この方法は手軽な割に効果が大きい。定跡の有無でレートが R100 位変わる。

「2 段仕込み定跡」を試している。これは、計算量と引き換えに上記の手法で作成した定跡に含まれている可能性のある悪手を排除する手法である。この手法の問題点は上記で登録された「対振り飛車定跡」が削除されてしまう点である。

最終的には「2 段仕込み定跡」か上記で生成した定跡との「ブレンド定跡」の良い方を使う予定。（対振り飛車には「ブレンド定跡」の方が良さそう。）

6.思考時間制御
手数と推定勝率に応じて思考時間を最大５倍まで変化させる様にした。

7.探索部

Hybrid 探索

２つの異なる学習を行った model(異なる極小値へ収束した可能性が高い)を用意し これらを A,B とする時 A=>A=>A または B=>B=>B の様に単独で使用し探索するよりも、A=>B=>A=>B の様に交互に探索する方が良い事を確認した。

V235(k=256,36b)と V252(k=384,24b)の２つの model を使って Hybrid 探索を行う。

8.高速化

network 構造の工夫、及び GPU が rtx4090・rtx3090 の場合に最適化した探索部の工夫により、同じパラメータ数を持つ ResNET 構造の model と dl 将棋標準の探索部の組み合わせに対して約２倍の高速化を達成した。

今回使用する予定の model:V235 と model:V252 の Hybrid 探索で、約 48000nps 程度の探索速度になった。

9.予想レート

以上の工夫により WCSC33 では R4200-R4300 程度だったが、今回は同じハードウェアで R4400-R4500 になったと推定している。

以下は少し細かな内容

10. Network 構造

Network を次の３つに分ける。

(1)データ入力部 (2)ResNET 部 (3)結果の出力部(Dual Head 部)

1）データ入力部

入力データの構成は標準と同じ。構造はオリジナルとは異なる。

2）ResNET 部

標準の ResNET とは異なる構造を採用。（ 以下 AstraNET と称する。）

WCSC33 では、カーネル数を k, ブロック数を b とする時、(k=256,b=5) + (k=224,b=20) の２段構造の通常の ResNET(合計 25 層)を採用した。

今年は、ResNET を AstraNET(自称)に置き換えた。また 20b~36b まで種々のサイズの model を作成し強さを比較した。

k=256 で 20b 及び 24b の model では思考時間を延ばしても R4400-4500 のソフトに読み負ける事を確認。これよりも重い Network が必要。

現時点では k=256,36b 及び k=384,24b の寸胴型の物を使う予定。NVIDIA 製 GPU のハードウェアの構造上２段構造にしても探索速度へのメリットは小さいと判断し寸胴型にした。

3）結果の出力部 (Policy/Value Network の分岐後)
dlshogi と同じ構造。
4）その他
AstraNET の活性化関数は主に ReLU を、全結合層には ELU を使っている。
(去年は全結合層に SiLU を使った)

11. 学習方法
１）学習率(lr)：0.2 から始めて 0.000003 まで等比数列的に減少させ学習した。減少させる割合は 0.80～0.95。d 値(=Loss-AverageAcc) の下がり方を見ながら調整した。
２）weight decay を 0.00010～0.00003 まで振って評価した。0.00003 は小さすぎる可能性がある。学習初期は大き目の物を使って学習終盤に小さくするのが良さそう。
３） 学習の途中で AstraNET のブロックの順序の入れ替えを適宜行っている。（去年も同様）

12. model の「追加工」
学習後に model の「追加工」をおこなった。
必ずしも Loss 値や d 値の最も小さい物、PolicyAcc,ValueAcc,Entropy の最も大きい物がベストな model とは限らない。
学習終盤では上記の指標を参考に良さそうな model を選択しベンチを実施。ベンチの結果の良かった複数の model のブレンド(例えば、3:2:1、100:1、100：-1 等)や乱数による揺らぎを与え派生 model を作成した。これらの中から最もベンチ結果の良かった物を最終的に採用した。
ベンチは棋譜生成も兼ねており次回の学習に使用する為 10k po~40k po で行った。
「追加工」により R+60 程度向上する場合も有った。

10. Optuna での探索パラメータの調整
去年の model は手作業である程度絞り込んでから Optuna を使った。
今回は 7 つある探索パラメータから以下の様なグループ A、B、C、D を作り、
A=>B=>C=>A(不十分な場合 B=>C=>A を追加)=>D=>E の様に調整した。
１回の Trial 数は 8～15 程度。全てを合計しても 100Trial は超えていない。

A: Softmax_Temp., root パラメータ 計 4 個
B: Softmax_Temp., non-root パラメータ 計 4 個
C: Softmax_Temp., C_fpu_reduction, C_fpu_reduction_root 計 3 個
D: 全パラメータ 計 7 個
E:Softmax_Temp., C_fpu_reduction, C_fpu_reduction_root を個別に振って妥当性を確認

一回の調整毎に上位 2~3 個のベンチを取り、ベンチ結果の最も良い物を次の中心値とした。Optuna での相手は Hao(3.5e6 nodes),ベンチでの相手は Hao(~6.0e6 nodes)と

Tanuki-dr4(~5.0e5 nodes)を使った。評価される側の勝率が 50~65%になる様に po 数を調整した。1 Trial の games 数は 250 とした。

11.中終盤の棋力向上

WCSC33 では、80~100 手付近で読み負けるケースが多かった。その対策として以下の 3 点を行った。

1) 自己生成棋譜は、去年は意図的に 40～80 手を重複させ 30 手～120 手の局面を学習データとした。121 手以降は未使用であった。今回は、意図的な重複を排した 30～150 手の範囲の局面を使用した。

2) 学習終盤では全学習データ内の重複局面を削除して学習した。これにより序盤～中盤初期の棋譜の割合が減る。

全データを毎回全て学習するのは時間が掛かりすぎるため、学習時間が 12 時間を超えると次の学習条件に移る様にした。経験的に学習条件を変更した直後から 3 ～ 4 時間の間に有望な model が得られる事が多く、逆に 8 時間以上同一条件で学習を続けても良い model が得られる事は少なかった。

3) 持ち時間の使い方を変更し、70～95 手の思考時間を延ばした。また思考時間を延長する場合、従来 2 倍までだった物を条件によって最大 4 倍まで延長するようにした。

-以上-

名人コブラ アピール文書

松山洋章

概要

ディープラーニング評価関数とNNUE評価関数をベースとしたアンサンブル評価関数を使用します

評価関数

複数のNNUE評価関数をマージして
利用します

定跡

意図的にランダムな手を混ぜた自己対戦により定跡を作成します

使用ライブラリ

- **dlshogi**

  局面評価のベースとして。

  序中盤の局面評価に優れるため。

- **やねうら王**

  局面評価のベースとして。

  読みの速さに優れるため。

ソフト名の由来

劇団鋼鉄村松

「二手目8七飛車成り戦法」

登場人物より

参考:
https://stage.corich.jp/stage_main/23036

第34回世界コンピュータ将棋選手権　「Lightweight」アピール文書

2024.3.31
神田 剛志

■開発動機

DL系単体での強さは各大会の結果にも表れてきてはいるものの、ハード側にそれ相応のスペックが必要なように見えます。
そのため、家庭用ローカルPCの範囲で、上位ソフト陣に比肩する棋力を獲得できることを示したい。名前の通り「軽く速く」が開発コンセプトです。

■アピールポイント/開発過程

①モデルアーキテクチャ

本家のResNetをEfficientNetで再構築し、１から学習しなおしました。
第3回電竜戦時のモデルからさらに層・チャネルを追加することでPolicyとValueともに精度を向上させています。
またEfficientNet単体ではなく、入力部に7層のResidual blockを入れ、そこからEfficientNetへ接続しています。

②USIエンジンのパラメータ設定変更によるNPS向上

GPUに局面を渡す際のバッチサイズを1024に上げています。
これと軽量なモデルと組み合わせにより、平均NPSを向上させています。

③GCT学習データによる教師あり学習とLightweight自身による強化学習

dlshogiチームが公開してくださっている学習データ（以下ⅰ,ii,iii）とLightweightの自己対局データを用いた強化学習を実施しています。

ⅰ. floodgateから抽出・作成された学習データ
ii. GCT電竜の自己対局データ
iii. 水匠による入玉局面データ
iv. 書籍「強い将棋ソフトの創りかた」に付属する学習データ
v. Lightweight自身による自己対局データ

④KL情報量による時間制御

dlshogi本家に倣い、Policyの確率分布と探索後の確率分布のKL情報量を用いた時間制御を導入しています。

⑤定跡の使用

Lightweightのモデルを利用し、初期局面の事前探索結果を利用することで、持ち時間の消費を抑えます。

⑥探索部の変更

PUCTアルゴリズムに従って探索木を降りていく際、各子ノードの着手確率を利用した簡易的な枝刈りを実施することで、最大UCB値の子ノード選択処理にかかる時間を短縮し、探索処理を効率化・高速化しています。

⑦MultiStream対応

dlshogi本家に倣いMultiStreamに対応することでNPSを向上させます。

⑧入力特徴量作成の改善

dlshogi本家に倣い入力特徴量の作成処理を改善し、NPSを向上させます。

⑨知識蒸留を用いたDNNモデルの精度向上

第3回世界将棋AI電竜戦に使用したDNNモデルを教師として
Lightweightのモデルを学習させています。

⑩合議制の採用

これまではDL系のみを使っていましたが、本大会ではNNUEとの合議を
採用する予定です。

■追試可否

可能。

■使用ライブラリ等

dlshogi：自己対局データ生成・探索部・モデル学習・定跡作成等に利用
Gikou2 ：検証時のテスト対局に使用
Suisho3：検証時のテスト対局に使用
Suisho4：検証時のテスト対局に使用
Suisho5：検証時のテスト対局に使用
elmo for learn：学習データ作成に利用

ponkotsu アピール文書

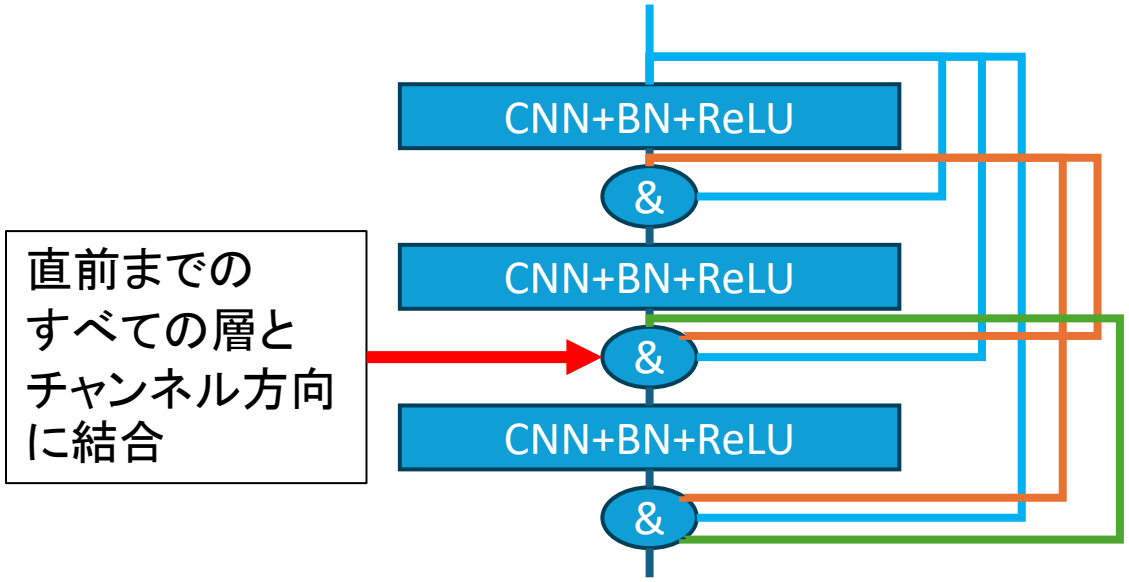
WCSC33からの改良内容

- ネットワーク構造の改良
- 学習率スケジューラの変更
- 最適化関数の変更
- 学習データのアップデート

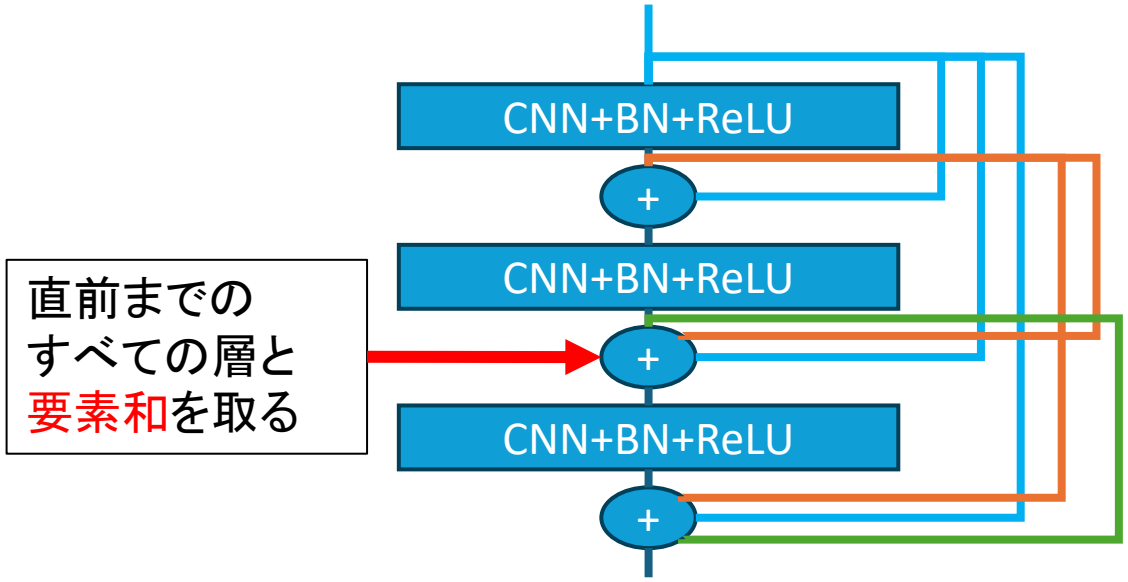
ネットワーク構造の改良

- 従来版で問題となっていた層が深くなるにつれチャンネル数が増大する問題の対策として、スキップ接続をチャンネル方向に結合するのではなく要素和を取るようにした

従来版(DenseNet)



改良版



※オーバーフロー対策として要素和を取った後平均を取っている

ネットワーク構造の改良(識別精度の比較)

- 学習データ: DL将棋本付属訓練データ
- epoch数: 22 epoch
- テストデータ: DL将棋本付属テストデータ
- Policy・Valueともに改良後のほうが高精度

Model	Policy Accuracy(%)	Value Accuracy(%)
DenseNet10	43.4	72.2
改良版DenseNet10	**44.3**	**72.9**

学習率スケジューラの変更

- 学習率スケジューラをReduceLROnPlateauに変更

ReduceLROnPlateauでは

$\mathrm{loss}_{\mathrm{best}}$: 最も良かったloss
threshold: 閾値
patiance: 改善が見られなかったepoch数

としたときlossがpatience epoch連続で

$$\mathrm{loss}_{\mathrm{best}} * (1 - \mathrm{threshold})$$

以上だった場合学習率を下げる

最適化関数の変更

- 最適化関数をSAM(Sharpness-Aware Minimization)に変更

以下の手順でパラメータを更新する

1. パラメータwの周辺で損失が最大となる$w + \hat{\epsilon}(w)$を算出
2. $w + \hat{\epsilon}(w)$における損失・勾配を算出
3. 2.で算出した勾配でwを更新

最適化関数の変更(識別精度の比較)

- 学習データ: DL将棋本付属訓練データ
- epoch数: SGD 22 epoch、SAM 11epoch
- テストデータ: DL将棋本付属テストデータ
- Policy・ValueともにSAMを使用したほうが高精度

Model	Policy Accuracy(%)	Value Accuracy(%)
DenseNet10+SGD 22 epoch	43.4	72.2
DenseNet10+SAM 11 epoch	**44.0**	**72.7**

学習データのアップデート

- DL将棋本の付属データのうちfloodgateの棋譜と水匠の棋譜を以下のようにアップデート

種類	従来の内容	アップデート後の内容
floodgateの棋譜	floodgateの2019年～2021年5月のレーティング3500以上のプレイヤーの棋譜	floodgateの2019年～2024年3月のレーティング3800以上のプレイヤーの棋譜
水匠の棋譜	水匠3改を使用して、1手120万ノードで自己対局した棋譜	たややんさん作成の最新の水匠を使用して1手1000万ノードで自己対局した棋譜(リンク)

その他

- 以上の改良に加え、ネットワークのブロック数を10→15に増やす予定
- 後日昨年のバージョンとレーティングを比較する予定

参考文献

- ReduceLROnPlateau — PyTorch 2.2 documentation (2024年3月27日閲覧)
- "Sharpness-Aware Minimization for Efficiently Improving Generalization", Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B. (2020)
- SoTAを総なめ！衝撃のオプティマイザー「SAM」爆誕&解説！ #機械学習 – Qiita (2024年3月27日閲覧)

お前、CSA 会員にならねーか？ アピール文書

ザイオソフト コンピュータ将棋サークル
野田久順 岡部淳 鈴木崇啓
河野明男 伊苅久裕

目次

- お前、CSA 会員にならねーか？
- 改良点
- 使用ライブラリ

お前、CSA 会員にならねーか？

- 奈川トモ先生の漫画『お前、タヌキにならねーか？』が元ネタです。

『お前、タヌキにならねーか？』をイメージして AI で生成した『お前、CSA 会員にならねーか？』のイメージ画像

改良点（1）

- nnue-pytorch を用いて学習しています。
  - nnue-pytorch は NNUE 評価関数の学習器です。
  - nnue-pytorch は、コンピュータチェス思考エンジン『Stockfish』の開発チームが開発しました。
  - コンピュータ将棋思考エンジンの NNUE の学習に対応させるため、やねうら王を組み込んでいます。
  - レーティングが向上し、学習時間も短くなりました。

改良点（2）

- Optimizer に Momentum SGD を使用しています。
  - Momentum なしで学習させた評価関数と比べ、R30.2 程度レーティングが高くなることを確認しました。

改良点（3）

- Suisho10Mn_psv を用いて Fine-tuning しています。
  - Suisho10Mn_psv は水匠シリーズ開発者 たややん様が公開されている学習データです。
  - 学習率を 1e-7 、λ＝0.0 で学習しています。
    - λ＝0.0 は勝敗項のみを見る設定です。

使用ライブラリ

- やねうら王
  - やねうら王を元に改造した思考部を使用しています。
    - 独自の工夫を加えるにあたり、改造しやすく、レーティングも高いためです。
- 水匠
  - 学習データを使用しています。
    - Fine-tuning により、レーティングが高くなることを確認したためです。
- tanuki-
  - 棋譜の生成に使用ています。
    - 過去に開発した資産の再利用のためです。
- nnue-pytorch
  - 評価関数の学習に使用しています。
    - レーティング向上と学習時間の高速化のためです。

大事なことは、CSA 会員になったら見つかるかも。

アピール文

花井祐　2024 年 3 月 14 日記

プログラム名：いちびん

プログラムの内容は以下のとおりです。

1.概要

NNUE を基本としています。思考部は、自作の教師データを使って学習を行っています。探索は、やねうら王のソースコードに手を入れて時間管理の部分を書き換えています。定跡は WCSC33 に参加したバージョンを基本に作成しています。

2.プログラムの内容

2.1　思考部分

今では少々時代遅れになってしまった NNUE 路線を守っています。教師データは、自己対戦で数年前に作成した深さ１２～１４の教師データを数億局面作成し、評価関数を作成しています。１年間、努力しましたが WCSC33 バージョンを上回る強さにはできなかったと考えます。

2.2　探索部分

WCSC33 で、比較的良い成績を残せたのは、時間管理を少し工夫したことだと自己評価しています。WCSC34 では、時間管理をさらにブラッシュアップしました。その内容は、局面の評価点がマイナス６００点より悪化した場合には、「ほとんど負け」として、それ以前に思考時間の山場をもってくるようにしています。

2.3　定跡

WCSC33（2023 年 5 月開催）終了後の翌日から、ほぼ 1 年間、機械を動かし続け、ひたすら定跡を作り続けました。「いちびん WCSC33 参加バージョン」を 1 手あたり１０億局面読ませる作業をつづけ、消費する電気代に涙しつつ、今日に至っています。

３　その他

WCSC33（2023 年 5 月開催）終了後に、DL について導入を考えましたが、現在のコンピュータ将棋にかける情熱と予想される高額な費用とを考えて DL への挑戦をあきらめました。頭を使いながら評価関数を手作りしていた時代は遠い過去のことになってしまいました。

以上

Polonaise アピール文書

谷合廣紀

2024 年 3 月 31 日

1　独自の工夫

基本は dlshogi/ふかうら王などのいわゆる DL 系で採用されている、policy+value Network + MCST のアプローチを取っています。dlshogi/ふかうら王と大きく異なるのは、モデル構造とその入出力です。

1.1　モデル入力のエンコード

モデルに盤面を入力するにあたって、まずは盤面情報を数値行列である入力特徴量に変換する必要があります。dlshogi では駒の位置や利きなどを 9x9 の 2 次元行列にエンコードしていき、最終的に 9x9x 特徴数の大きさを持つ入力特徴量を得ています。この入力特徴量は CNN を使い推論されていくため、dlshogi は画像処理的なエンコードと捉えることができます。

一方の Polonaise では、盤面を 1 一から順に見ていき、1 一、1 二 ･･･ 9 九の駒と先後の持ち駒（7 種 x2）を並べた 95 字の文字列にエンコードすることで入力特徴量を得ます。この入力特徴量はモデルの最初の層で埋め込み層によりベクトルに変換されて推論されていくため、自然言語処理的なエンコードと捉えることができます。

具体的に Polonaise のエンコード方法を図1の盤面を使って示します。

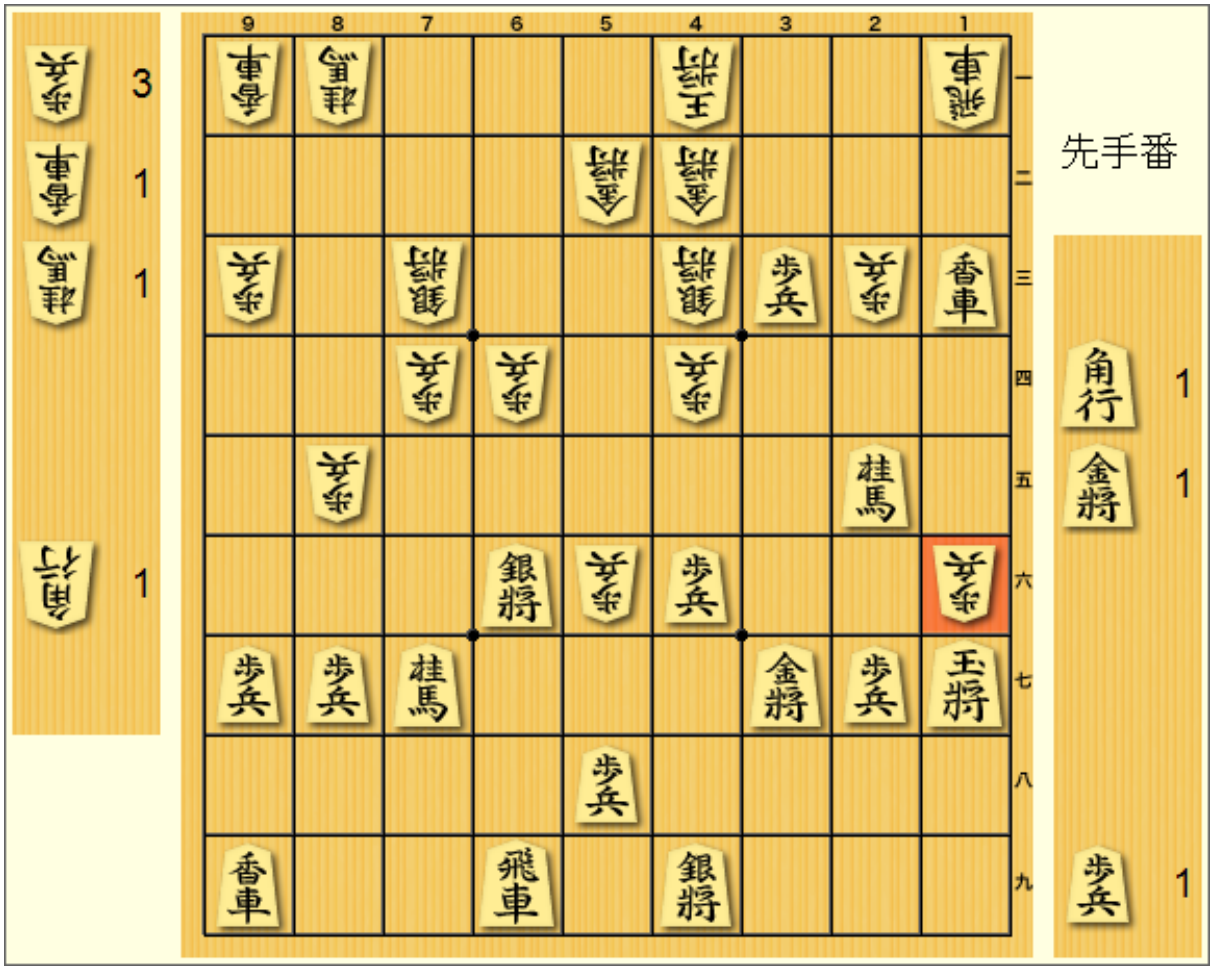


図1: 入力盤面

この盤面の駒を 1 一から順に見ていくと、「後手の飛車」、「空きマス」、「先手の香車」･･･ と続きます。また、持ち駒は先手は歩が 2 枚、金が 1 枚、角が 1 枚です。盤面 81 マスの情報はそのまま駒情報が入り、持ち駒はそれぞれの駒を何枚持っているかが入ります。したがって図1をエンコードすると、図2の文字列が得られます。

実際には文字列だと扱いづらいため、各文字が対応する数値 ID に変換されて、95 個の数値列がエンコードされた入力特徴量となります。

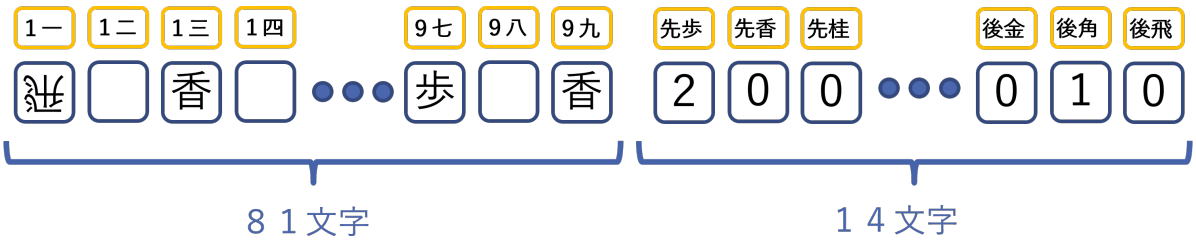


図2: 文字列エンコード

1.2　モデル構造

95 個の数値列は、最初の埋め込み層によって 95x256 の大きさの行列に変換されます。これまでは gMLP[?] と呼ばれる MLP ベースのモデルを採用していましたが、今回は BERT-large 程度の中規模なモデルを採用しています。

1.3　モデル出力

dlshogi で採用されている policy の出力は、「着手するマス」と「その駒はどの方向から来たか」の組み合わせで表現されます。「着手するマス」は 81 マスあり、「どの方向から」は 27 通りあるため、その組み合わせは 2187 通りです。したがって policy は 2187 通りのクラス分類問題として表現されています。

しかし、「1 一のマス」に「下がる」や「左に寄る」といった動きは将棋の合法手として存在しません。このように dlshogi の policy 表現の中には決して現れない組み合わせがいくつかあります。それら非合法手を数え上げていくと 691 通りあり、約 32% が非合法手となっていることがわかります。

実験の結果、policy の出力を 2187 クラスの分類問題として解くよりも、非合法手 691 通りを除いた 1496 のクラス分類問題として解いた方が、policy の学習がうまくいくことがわかりました。そのため Polonaise では 1496 のクラス分類問題として policy の学習を行っています。

2　使用ライブラリ・使用データ

- ふかうら王
- 「強い将棋ソフトの創り方」公開データ

参考文献

Daigorilla WCSC34 アピール文章・・・急いで書きました泣

工夫した点

① 後手番勝率の強化：　後手は角換わりを避け、雁木もしくは力戦にさすように model のパラメータを調整した。

② 探索部の高速化及びパラメーターの調整:ふかうら王を CUDA12.3　TensorRT8.6、cudnn9.0 に最適化し、後手の勝率を疑似調整した棋譜で自動調整を行った。ちなみにやねさんのスポンサーには入っていないのでバージョンは 8.00

③ Model の最適化：特に至って工夫はしていないが 20b_swish で学習を進めている。現時点では手動と自動で調整している定跡を用いて標準型 NNUE 関数「Hao」を用いて depth14～18、約 5 億局面を学習させている。学習率を徐々に下げている。

④ 定跡の強化：秘密にしておきたいが上記の通りかなり手間をかけて生成している。無駄な局面を減らすため実現確立が高いものから局面を誘導し、様々な評価関数に対応できるようにしている。詳しくは決勝に残ったら書きたい。

Team Novice

wcsc34

Update..

・昨年までDL部を中心に開発していた中屋敷さんが別ソフトとして出場します

　→　nshogi として出場しています！

・これに伴い、昨年のバージョンからDL部を切り離し、NNUEのみでの出場となります

・NNUE部は昨年から開発が止まっているため、目安ソフト的な立ち位置として出場します

・定跡は例年通り新たに作成する予定です

・フルスクラッチについては、開発が止まっていることもあり本年は申請していません

WCSC34 koron アピール文章

野田煌介

2024/03/28

1 前回までの課題

現在、NNUE 系の将棋 AI では表現力の限界と序盤-終盤の棋力トレードオフが課題になっていると考えられる。

以前の koron はこれらの課題に対して、「ある一定の手数を経過した場合に評価関数を切り替える」といった手法をしばしば採用してきた。

しかしながら、大会結果からも分かる通りこれらは大した成果を上げることは出来なかった。

原因として、局面がどのような状況であるにも関わらず、手数で序盤終盤を判断し、評価関数を切り替えるといった手法に問題があったと考えた。

2 改善点

そこで今回は評価値に応じて評価関数を切り替える手法に変更した。

一定の評価値以内では重いネットワークを使用し、評価値が有利不利どちらかに偏った場合、軽いネットワークを使用する。また評価値に関係なく、どちらかが入玉した場合も軽いネットワークに変更する。

これにより、序盤では精度の向上、終盤では読み抜けの防止や入玉将棋におけに棋力の向上などを両立できると考える。

3 使用ライブラリ

やねうら王

　　優秀な思考エンジンであるため。

nodchip 氏が公開している教師データ群

　　質、量ともに優秀な教師データであるため。

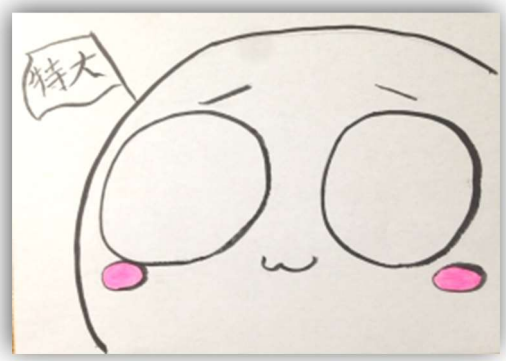
たややん氏が公開している教師データ群

　　Fine Tuning において優秀な教師データであるため。

TMOQ アピール文書

2024 年 03 月 20 日 作成
2024 年 04 月 30 日 改訂

【ソフト名】TMOQ（特大もっきゅ）
“TMOQ”と書いて「特大もっきゅ」と呼びます、
愛娘が命名しデザインしたものです



【コンピュータ将棋大会実績】
2016 年の WCSC26 以降、ほぼ全ての大会に出場、
そこそこの成績でやってきてます

【WCSC34 の目標】
少なくとも 1 回は入玉宣言勝ちする！

【TMOQ の特徴 および WCSC33 との違い】

1. 戦いを避ける平和主義者（入玉宣言勝ち狙い）
2. dlshogi をベースに複数エンジンを使用
3. メインの思考部は『Lí（Ponder 無し)』(昨年は ResNet 9b(Ponder 有り))
4. 勝勢になったら、駒得エンジンに切替えて入玉を目指す（入玉可能性を増やすべく、昨年の駒得エンジンをチューニング）
5. 約 2 千 5 百万手に近い定跡ファイル（昨年より約 300 万手増）
6. Note PC を使用、莫大な計算資源がなくてもコンピュータ将棋は楽しめる！

※ 必ずしも強くなることを目指している訳ではありませんが、強豪ぞろいの二次予選で入玉宣言勝ちすべく、メインの思考部を独自の ResNet 9b から『Lí』に切り替えました。結果、昨年 WCSC33 版とのテスト対局で約 76 %の勝率となりました

【使用ライブラリ】

- 『DeepLearningShogi』(Commit 790e2f4 on 3 Feb 2022) (GPL)
- 『Lí』（tanuki- 第 33 回世界コンピュータ将棋選手権バージョン）
- 『やねうら王』 V7.00

【御礼】

今回も山岡氏、加納氏、野田氏＆やねうらお氏を中心に、多くの方々の公開情報により参加できました。この場を借りて御礼申し上げます

AobaZero の 2024 年のアピール文書

山下 宏
yss@bd.mbn.or.jp

1 AlphaZero の追試が最初の目的

AobaZero は Bonanza、LeelaZero のコードをベースに AlphaZero の追試をするべく MCTS ＋ディープラーニングで実装されてます。ネットワークは 3x3 のフィルタが 256 個の 20 block の ResNet でパラメータの個数は 2340 万個。棋譜生成をユーザの皆様と協力して行う分散強化学習です。オープンソースです[*1]。

2 AlphaZero の追試は 2021 年 4 月に終了

AlphaZero の将棋の追試は、2019 年 3 月から開始し、2021 年 4 月に 3900 万棋譜を作成して終了しました[*2]。2024 年 3 月 29 日現在、6746 万棋譜を作成しています。

3 追試終了後から +260 ELO、去年の選手権から +30 ELO

追試終了後からは +260 ELO、去年の選手権からだと +30 ELO ほど強くなっています。追試終了時では AlphaZero より +150 ELO 弱い、という推定でしたが現在は +260 なので AlphaZero を +110 ELO 程度、超えた棋力かもしれません。

4 昨年の選手権からの主な変更点

- 学習棋譜の平均 playout 数を 1600 から 3200 に変更
- 先手勝ち局面の学習確率を減らす
- NVIDIA のドライバ更新時のエラー修正

4.1 学習棋譜の平均 playout 数を 1600 から 3200 に

ほぼ重みの棋力が停滞していたので playout 数を倍にしてより強い棋譜を生成するようにしました。当然生成時間は倍かかるので、生成棋譜数も 11000 棋譜/日から 9700 棋譜/日に下がってます。・・・ほとんど下がっていないですね。実は倍に変更してからかなりの資源を投入して棋譜生成に協力してくださった方がいたためです。この場をお借りして感謝いたします。棋譜の棋力はほぼ同じ条件で floodgate で走らせた限りでは平均 1600 の 3630 から平均 3200 で 3740 と +110 ELO 程度向上しているようです。この平均、とは 400playout ごとに Root の着手の訪問数の分布に変化がないなら打ち切る、という手法です。最小 400、最大 12800 です。この KL 情報量を利用した探索の打ち切りは LeelaChessZero で使われています[*3]。この探索量の調整で 1 手 3200 固定より +100 ほど強くなります。

4.2 先手勝ち局面の学習確率を減らす

playout 数を倍にして棋力を上げたのですが、生成棋譜の先手勝率が 0.724 と 7 割を超えていました。PUCT の性質で勝率が悪い局面だと多くの候補手を調べて、さらに弱くなってるケースがありそうです。そこで先手が勝った局面の学習確率を減らして初期局面の勝率が 5 割になる重みを再学習で作ってみたところ、棋力も +14 ELO 程度強いものが出来たので、これに差し替えました。その結果、先手勝率は 0.632 と 1 割近く下がりました。形勢判断としては正しくないですが棋譜生成においてはこちらの方が望ましいかもしれません。

4.3 NVIDIA のドライバ更新時のエラー

NVIDIA のドライバを最新にすると AobaZero が起動時に動かなくなる、という現象が発生しました。これは初回起動時に OpenCL の最適な設定を見つける行列計算を行う際に、OpenCL を動的にコンパイルしていたのですがコンパイラのエラーチェックが厳しくなったせいでした。芝さんなどの指摘で気づきました。ありがとうございます。

5 dlshogi 互換モデルで出場予定

AobaZero の棋譜で dlshogi のモデルを作ってみました。AobaZero と同じ 20 ブロック、256 フィルタの ResNet、Swish だと学習速度は 5.7 倍！探索速度の NPS は 3 倍ほど速いです。学習は 3090 だとミニバッチ 256 が最大だったのですがミニバッチ 2048 もあっさり動いたのにビックリでした。

1 手 1playout と 1 手 100playout、nps(RTX 3090) は表 1 です。ミニバッチはすべて 1 です (nps 測定では AobaZero が mb=17、dlshogi が mb=256)。dlshogi dr2 には Policy の強さ (1p)、探索 (100p) でもまだはっきり負けています。重みサイズが 2 分の 1 程度にも関わらず。384x30b にしてもまだ Policy の精度で負けてるので何か学習が正しくない気はします。ただ実時間だと nps が 3 倍近いのもあり (3090

[*1] https://github.com/kobanium/aobazero
[*2] https://github.com/kobanium/aobazero/issues/54
[*3] https://lczero.org/dev/wiki/technical-explanation-of-leela-chess-zero/

で AobaZero は 6000 程度) +40 ELO 程度 dlshogi モデルが強いので、これで選手権は参加する予定です。AobaZero は TensorRT に対応してないのと、複数 GPU での性能がいまいちなのもあって。計測は 24 手までの互角局面集利用で先後入れ替えて 800 局からです。

5.1 局面選択は Value に効果的？

学習は通常の AobaZero の学習と同じように学習される局面を選択しています[*4]。

先手勝率を 5 割になるように先手勝ち局面の選択割合を減らす、序盤の学習確率を減らす、勝率が 80 %程度の手が 8 倍程度で学習されやすく、など。全局面の 50 %ほどが選択されるぐらいで。それを csa 形式に変換した後、aoba_to_hcpe3.py を改造して hcpe3 形式に変換しています。この選択をなしで hcpe 形式 (着手のみ) で学習させた場合は Policy の精度はいいのですが 1 手 100playout で 251 差、とはっきり弱く、局面の選択をするのは Value の精度に効いてるかもしれません。なお重複局面を平均化して削除する dlshogi の学習機能はどちらも有効にしています。

学習は 4000 万から 6690 万までの 2690 万棋譜から 5 億 7000 万局面を抽出してミニバッチ 2048 で学習率 0.1 から 0.001 まで Cosine Annealing で学習させました。さらに 0.00001 ぐらいまで下げた方がもう少し性能が上がるようです。

表 1　AobaZero(w4357) の dlshogi モデルに対する ELO

	1p/手	100p/手	nps
dlshogi dr2 (224x15b)	-61	-100	25300
AobaZero 256x20b	186	100	20500
AobaZero 256x20b(選択なし)	157	251	20500
AobaZero 384x30b	61	-58	5500

6　定跡は floodgate の AobaZero の棋譜から

定跡は floodgate で走らせている AobaZero の棋譜の局面を 30 万局面ほど探索させて作成する予定です。

7　対振り飛車の弱さが課題

floodagte で HoneyWaffle との棋譜を見ると相手が飛車を振っただけで評価値が +400 (勝率 65 %) と大きく出ます。学習棋譜でも振り飛車の割合は 2 %しかなく、ミレニアムや振飛車穴熊、といった囲いの知識もないので多様性に欠ける感じはします。この影響もあり特に対振り飛車は弱い感じです。

8　人間の知識は使っていない、をおそらく継続

利きの情報の追加や 3 手詰、dfpn 詰などで AlphaZero からは離れてきましたが、まだ全体としては「人間の知識は使っていない」を継続していると考えています。

9　棋力の推移

図 1 が ELO の推移です。floodgate での測定レートの方が若干高めなのは Kristallweizen での棋力測定に用いている互角局面集 (24 手まで) には AobaZero が指さない穴熊や振飛車が多数含まれているせいです。局面集を使わずに初手から指させた方が +100 ELO ほど強くなります。

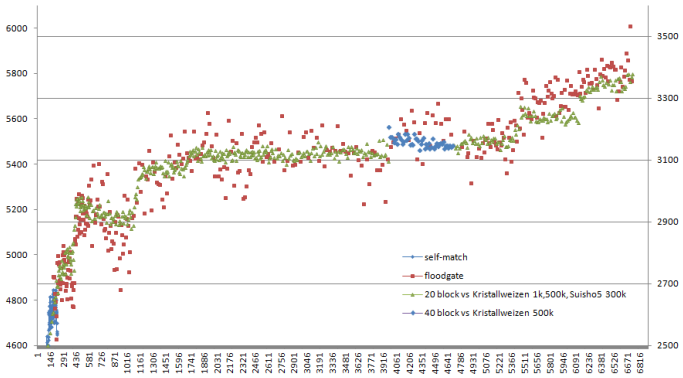


図 1　棋力の推移。右軸が floodgate のレート、横軸は棋譜数 (万)

10　5 年で 6700 万棋譜

6700 万棋譜、という膨大な棋譜を 5 年間で生成してきました。棋譜生成に協力していただいてる皆様に感謝いたします。

[*4] https://www.apply.computer-shogi.org/wcsc33/appeal/AobaZero/2023csa_appeal.pdf

十六式いろは煌（きらめき）

第 34 回世界コンピュータ将棋選手権
アピール文

メンバー

末吉竜介、（増える予定）

「十六式いろは 煌」の由来

昨年 2022 年に 2 期生の先輩達が決めた「十六式いろは煌 (きらめき) 」を今年も引き継ぎます。

以下、 2022 年当時の由来の記載です。

様々な名前の候補が上がり最終的に決まったのが考え始めてからなんと 1 か月かかりました！。
皇 (すめらぎ) 、煌 (きらめき) 、日本工学院、かまトゥ（学校のマスコットキャラ）…などなど。
「日本工学院の名前があった方がよいのではないか」や
「ローマ字で書いた方がかっこいい！」などかなりの意見などがありましたが
最終的には末吉先生の「十六式いろは」と生徒達で考えた「煌 (きらめき) 」を
組み合わせて決定しました！

ソフトの概要

採用予定

- やねうら王
- dlshogi
- KomoringHeights
- Electron 将棋

ソフトの説明（予定）

- やねうら王での評価関数は第４回電竜戦の
「十六式いろは煌（きらめき）」内部の
「十六式いろは幻（まほろ）」を改良したもの。
探索速度重視で標準 NNUE から軽量化する予定
- dlshogi のネットワークモデルは昨年に引き続き軽量なもの
（GhostNet・ゴーストネット）を採用
- 定跡ファイルは floodgate、wcsc、電竜戦、AobaZero の棋譜を
利用して作成する
- 詰将棋エンジン（KomoringHeights・コーモリンハイツ？）を
含めて合議制の見直し

2024.03.31

wcsc34 での実装について

（次のページから記します）

基本的な動作

―昨年 2022 年の wcsc32 と同様。

やねうら王（★１）と dlshogi （★２）、詰将棋エンジンの各エンジンによる合議がこのソフトの最大の特徴。 Ayane を使用して両エンジンを呼び出し、評価値を比較して指し手を決める。

★ １　評価関数は、第 4 回電竜戦の「十六式いろは煌（きらめき）」内部の評価関数「十六式いろは幻（まほろ）」（やねうら王の評価関数）を改良したもの。
★ ２　ネットワークモデルは第 4 回電竜戦の「十六式いろは煌（きらめき）」内部のネットワークモデル「十六式いろは幽（ほのか）」（ dlshogi のネットワークモデル）

wcsc33 時との違い

1) やねうら王の評価関数の変更（十六式いろは幻（まほろ））
2) dlshogi のネットワークモデルの変更（十六式いろは幽（ほのか））
3) 定跡ファイルの変更（内容は 03/31 現時点では未定）
4) 詰将棋エンジンを追加

変更１）やねうら王の評価関数の変更

第 4 回電竜戦での「十六式いろは幻（まほろ）」（★）から
さらに自己対戦で作成した教師局面データで追加学習したもの。
探索速度の向上を目的にし標準 NNUE ではなく軽量化した NNUE にする予定。

（★）十六式いろは幻（まほろ）
既存の教師局面データを使わず、自己対戦と学習を繰り返して
合計 50 億局面の教師局面データから学習したものに、
水匠 5 、 Hao 、 BLOSSOM で補完し、さらに自己対戦＆追加学習したもの。

変更２） dlshogi のネットワークモデルの変更

「十六式いろは幽（ほのか）」

探索速度の向上を目標に、軽量化と精度維持を目指したネットワークモデル。

具体的には、精度を維持しつつパラメータ数を大幅に削減し軽量化した
GhostNet （ゴーストネット）をメインに、
SENet （エスイーネット）、 Bottleneck （ボトルネック）を追加したもの。

変更３）定跡ファイルを新規で作成

WCSC、電竜戦、 AobaZero、 floodgate の棋譜を元に作成する予定。
方針は現時点（2024-03-31）では未定。

変更４）詰将棋エンジンを追加

詰将棋エンジン KomoringHeights も搭載する予定。

棋力（wcsc33 後に floodgate で測定）

十六式いろは煌（きらめき）　wcsc33（一次予選　7 位）　：　R3524
　マシン： GALLERIA XF （Core i7-9700K , GeForce RTX 2070 SUPER）

以下、内部エンジン単独
　十六式いろは幻（まほろ） - 第 4 回電竜戦：
　　VS Hao-wcsc33　：R-123.8
　　VS Suisho5　　：R-112.9
　十六式いろは幽（ほのか）：　未測定

使用ソフト、参考文献等、その１

sueyoshiryosuke/16shiki-Iroha_kirameki:
https://github.com/sueyoshiryosuke/16shiki-Iroha_kirameki
←wcsc32 時のもの。

TadaoYamaoka/DeepLearningShogi
https://github.com/TadaoYamaoka/DeepLearningShogi

yaneurao/YaneuraOu
https://github.com/yaneurao/YaneuraOu

Electron 将棋
https://sunfish-shogi.github.io/electron-shogi/

Home · yaneurao/YaneuraOu Wiki
https://github.com/yaneurao/YaneuraOu/wiki

tttak/GougiShogi: 合議将棋
https://github.com/tttak/GougiShogi

使用ソフト、参考文献等、その２

オープンソースの詰将棋エンジン「KomoringHeights」を作った・コウモリのちょーおんぱ
https://komorinfo.com/blog/komoring-heights/

yaneurao/Ayane:
https://github.com/yaneurao/Ayane

AobaZero
http://www.yss-aya.com/aobazero/

floodgate
http://wdoor.c.u-tokyo.ac.jp/shogi/floodgate.html

次世代の将棋思考エンジン、 NNUE 関数を学ぼう（その１．ネットワーク構造編）
- コンピュータ将棋 Qhapaq
https://qhapaq.hatenablog.com/entry/2018/06/02/221612

tanuki- 2022-06-07 やねうら王学習部リグレッション調査 - nodchip のブログ
https://nodchip.hatenablog.com/entry/2022/06/07/000000

使用ソフト、参考文献等、その３

人間の棋譜を用いずに評価関数の学習に成功 | やねうら王 公式サイト
https://yaneuraou.yaneu.com/2017/06/12/%E4%BA%BA%E9%96%93%E3%81%AE%E6%A3%8B%E8%AD%9C%E3%82%92%E7%94%A8%E3%81%84%E3%81%9A%E3%81%AB%E8%A9%95%E4%BE%A1%E9%96%A2%E6%95%B0%E3%81%AE%E5%AD%A6%E7%BF%92%E3%81%AB%E6%88%90%E5%8A%9F/

lambda 混合絞りについて | やねうら王 公式サイト
https://yaneuraou.yaneu.com/2017/08/21/lambda%E6%B7%B7%E5%90%88%E7%B5%9E%E3%82%8A%E3%81%AB%E3%81%A4%E3%81%84%E3%81%A6/

テラショック定跡の生成手法 | やねうら王 公式サイト
https://yaneuraou.yaneu.com/2019/04/19/tera-shock-book-generation/

ShogiGUI
http://shogigui.siganus.com/

ai5/BookConv
https://github.com/ai5/BookConv

使用ソフト、参考文献等、その 4

ak110/Blunder.Converter: 棋譜変換ツール。
https://github.com/ak110/Blunder.Converter

各種将棋ソフト間での教師データの変換ツールの開発 - コンピュータ将棋 Qhapaq
https://qhapaq.hatenablog.com/entry/2017/12/25/002820

たややん /ToSfenpack20210122
https://twitter.com/tayayan_ts/status/1338443561272950787?s=20

将棋 AI の進捗 その 31(cuDNN による SENet の推論処理の実装）
- TadaoYamaoka の開発日記
https://tadaoyamaoka.hatenablog.com/entry/2019/07/24/011150

強い将棋ソフトの創りかた | マイナビブックス
https://book.mynavi.jp/ec/products/detail/id=126887

精度を維持したままパラメータ数を大幅に削減「 GhostNet 」 | AI-SCHOLAR | AI ：（人工知能）論文・技術情報メディア
https://ai-scholar.tech/articles/image-recognition/ghostnet-ai-383