INUGAMI
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

- INUGAMI
- 改良点
- 使用ライブラリ

INUGAMI

- 『松山騒動八百八狸物語』に登場する化け狸が元ネタです。

『隠神刑部』をイメージして AI で生成した
『INUGAMI』のイメージ画像

改良点（1）

- 評価関数の学習に知識蒸留を用いています。
  - 知識蒸留は教師モデルの知識を生徒モデルに伝達する手法です。
  - Response-based Knowledge のオフライン知識蒸留を行っています。

改良点（2）

- 知識蒸留の具体的な手順は以下の通りです。
  1. CPU エンジン同士で自己対戦し、学習データを生成する。
     - 学習サンプルには局面と評価値、勝敗等を含む。
  2. 学習データの局面を GPU エンジンの評価関数に推論させ、勝率を出力させる。
  3. 勝率を評価値に変換する。
  4. 学習サンプルの評価値を 3. で置き換える。
  5. 4. を用いて NNUE 評価関数を学習する。

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
  - DL 水匠を使用しています。
    - GPU エンジンの評価関数の中で知識蒸留をした際に最もレーティングが高かったためです。
- 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。

『松山騒動八百八狸物語』のあらすじを読み、
『隠神刑部』の境遇が可哀そうすぎて悲しくなりました。

第 35 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
2025/3/20

1　dlshogi のアピールポイント

1.1　モデル構造

ResNet と Transformer を組み合わせた構造を採用する。

Transformer は、局面全体を見たグローバルな特徴を捉えることができる。しかし、局所的な特徴を学習するには比較的多くのデータ量が必要になり、ResNet と比べて学習の効率が悪い。

一方、ResNet は、構造上、入力に近いブロックで局所的な特徴を捉えやすい帰納バイアスを持っているため、局所的な特徴を学習する効率が良い。

そこで、Transformer を ResNet の後ろに接続することで、ResNet が効率的に捉えた局所的な特徴量を使って、Transformer で効率良くグローバルな特徴を捉えることができる。

このような ResNet と Transformer を組み合わせた構造は、画像認識の分野では、「Bottleneck Transformers for Visual Recognition[1]」などで提案されている。

1.2　ラージカーネル

ResNet は、複数の層を重ねることで、グローバルな特徴を捉えることができる。入力に近い層では局所的な特徴を捉えて、後続の層で、徐々にグローバルな特徴を捉えられるようになる。

以前より、カーネルサイズを大きくすることで、少ない層で、グローバルな特徴を捉えられることが知られている。しかし、現在の GPU は 3x3 のカーネルサイズに最適化されており、計算速度の観点から大きなカーネルサイズは採用されていない。

「A ConvNet for the 2020s[2]」では、カーネルサイズを大きくすることで、最先端の Vision Transformer と同等の性能が出せることが報告されている。

「InceptionNeXt[3]」では、大きなカーネルを、縦と横に分割することで、計算コストを抑えて、同等の性能が出せることが報告されている。

第 33 回世界コンピュータ将棋選手権の、Ryfamate が採用した「Ryfamate Cross Network[4]」は、9x9 のカーネルを 9x1 と 1x9 のカーネルに分割する構造を採用している。将棋の盤面が

[1] https://arxiv.org/abs/2101.11605
[2] https://arxiv.org/abs/2201.03545
[3] https://arxiv.org/abs/2303.16900
[4] https://www.apply.computer-shogi.org/wcsc33/appeal/Ryfamate/appeal_ryfamate_20230421.pdf

9x9 固定であるため、ストライドが不要になり、大幅に計算コストを下げられる利点がある。

実験[5]したところ、Ryfamate Cross Network は、7x7 のカーネルよりも、効率的に高い精度が出せることが分かった。しかし、ResNet から置き換える層を増やし過ぎると、逆に精度が下がる現象も確認できた。これは、9x1 と 1x9 のカーネルの出力がそれぞれ、1x9 と 9x1 になるため、元のサイズに戻すためにブロードキャストを行う過程で、位置情報が失われているためと考えた。そこで、1x1 のカーネルを並列で加えて実験したところ、層を増やしても精度が劣化しないことが確認できた[6]。

以上の結果を元に、局所的な特徴を捉える特性を残して、グローバルな特徴も位置情報を残しながら伝えられるように、通常の ResNet ブロックを 5 ブロック間隔で 9x1、1x9、1x1 のカーネルに置き換える構造を採用する。

1.3 入玉特徴量

コンピュータ将棋の大会で採用されている 27 点法では、先手と後手で宣言勝ちの条件が異なるため、モデルに手番の情報を入力しない限り、正しく局面を評価できない。

しかし、ほとんどの局面では、手番を対称に捉えても問題ないため、後手の場合、反転した局面を学習して、手番を無視して評価する方が効率が良い。

最近の大会では、入玉宣言で勝敗が決まる対局が増える傾向にあり、入玉の精度が勝敗に影響してきている。

そこで、より正確に入玉の状況を評価できるように、入力特徴量を追加することを検討した。

前述の通り、手番そのものを入力特徴量に加えるのは、入玉に無関係の局面での対称性が失われるため、あとどれくらいで入玉宣言勝ちできるかを逆算した特徴量として与えることにする。これにより、先手、後手の違いを吸収し、対称性を維持できる。

具体的には、以下の特徴量を追加した。

- 入玉しているか
- 敵陣の玉を除く枚数(10 枚までの残り枚数。ワンホットで与える。)
- 残り点数(先手は 28 点、後手は 27 点までの残り点数。19 を上限として残り点数をワンホットで与える)

同じ訓練データを入玉特徴量の有無で学習して、強さを計測したところ、入玉特徴量を加えた方が強くなることが確認できた[7]。入玉宣言勝ちの回数は増えなかったため、入玉するかの見極めが上手くなったと考えている。

1.4 モデルサイズ

これまでの dlshogi では、モデルのパラメータ数を増やすほど、同一持ち時間でも強くな

[5] https://tadaoyamaoka.hatenablog.com/entry/2024/07/20/160442
[6] https://tadaoyamaoka.hatenablog.com/entry/2024/07/26/224144
[7] https://tadaoyamaoka.hatenablog.com/entry/2024/12/29/141617

ることが確認できている。

前大会の 30 ブロック 384 フィルタのモデルのパラメータをさらに増やして、40 ブロック 512 フィルタの約 2 億パラメータのモデルを学習した。

訓練データは同一ではないが、40 ブロック 512 フィルタのモデルが、elo レーティングで 37.9 だけ強くなった[8]。

また、50 ブロック 640 フィルタの約 4 億パラメータのモデルの学習も行ったが、floodgate の棋譜に対する方策、価値の精度はともに大きく向上したが、探索速度が 53%程度となり、同一持ち時間では弱くなることが確認できた（GPU に H100 PCIe を使用）。

現状の GPU では、50 ブロック 640 フィルタのモデルは推論速度が大きく低下するため、探索が浅くなり過ぎて、同一持ち時間で弱くなったと考える。なお、同一探索数では強くなる。

パラメータ数を増やすほど同一持ち時間でも強くなるというスケーリング則は、50 ブロック 640 フィルタあたりで、成り立たなくなることがわかった。

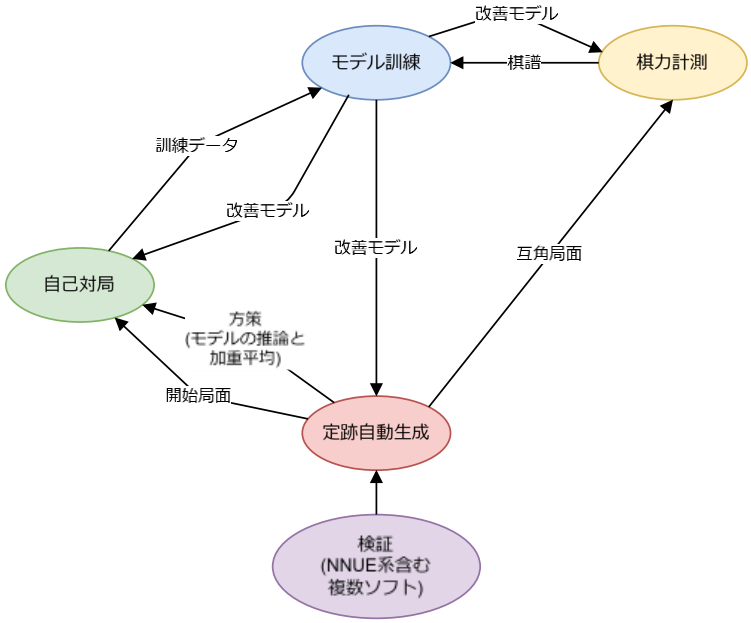
ただし、モデル精度は高いため、知識蒸留することで、活用することはできる。

1.5　定跡自動生成

前大会の定跡自動生成の手法[9]で、継続して定跡の自動生成を行った。

約 300 万局面が登録されている。

また、自動生成した定跡を、訓練データ生成、棋力測定にも活用している[10]。



[8] https://tadaoyamaoka.hatenablog.com/entry/2024/06/07/224348

[9] https://www.apply.computer-shogi.org/wcsc34/appeal/dlshogi_with_HEROZ/dlshogi_with_HEROZ_appeal_detail_wcsc34_v2.pdf

[10] https://tadaoyamaoka.hatenablog.com/entry/2024/12/22/194641

2　dlshogi の特徴

以下に、これまでの dlshogi の累積的な特徴を示す。

※下線部分は、第 34 回世界コンピュータ将棋選手権からの差分を示す。

- 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

3　使用ライブラリ

- Apery[11]（WCSC28）
  →局面管理、合法手生成のために使用

3.1　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。

4　各特長の具体的な詳細（独自性のアピール）

4.1　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。

4.2　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 36 ブロック(畳み込み 2 層で構成)と Transformer 2 ブロック(Transformer は Attention ブロックと FFN ブロックがあるため ResNet2 ブロック換算)で構成した。通常のカーネルサイズは 3x3（入力層の持ち駒のチャンネルのみ 1)、5 ブロック間隔で 9x1 と 1x9 と 1x1 を並列にしたカーネルに置き換える。フィルタ数は 512 とした。

4.3　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。

4.4　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

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

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。

4.5　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[12]の疑似コードに記述された式を使用した。

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。

4.6　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)。

4.7　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

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

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。

4.8　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[13]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。

4.9　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。

4.10 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。

4.11 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

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

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。

4.12 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。

4.13 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。

4.14 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。

4.15 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(4.27 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。

4.16 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。

4.17 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(4.13 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。

4.18 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。

4.19 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。

4.20 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。

4.21 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。

4.22 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。

4.23 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。

4.24 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。

4.25 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。

「4.7 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。

4.26 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[14]。

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。

4.27 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[15]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[16]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。

4.28 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。

4.29 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[17]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

---

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

[15] https://optuna.org/

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

[17] https://tadaoyamaoka.hatenablog.com/entry/2020/05/05/181849

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。

4.30 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[18]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。

4.31 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。

4.32 Stochastic Multi Ponder

相手番に、相手番の局面から探索を行う Stochastic Ponder（4.28 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[19]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 したエンジンに割り当てることで、前回の探索結果を再利用する。

[18] https://www.apply.computer-shogi.org/wcsc31/appeal/Qugiy/appeal_210518.pdf
[19] http://id.nii.ac.jp/1001/00199872/

第35回世界コンピュータ将棋選手権 「東横将棋」アピール文書

2025.3.31

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（nnue-pytorch等）に加え独自のNNUE評価関数の強化

・手作業による定跡の生成。今回は角換わり定跡に主眼を置いています。

・探索部はやねうら王を予定しています。いわゆるやねうら王チルドレンです。

よろしくお願いいたします。

んんんｗｗｗ

そうですなｗｗｗ我は「結果」だけを求めてはいないんですなｗｗｗ
「結果」だけを求めていると人は近道をしたがるものですなｗｗｗ
近道した時、真実を見失うかもしれませんなｗｗｗ
そしてやる気もしだいに失せていくかも知れませんぞｗｗｗ

大切なのは『真実に向かおうとする意志』だと思っていますぞｗｗｗ

向かおうとする意志さえあれば、たとえ思うような結果が残せなくてもいつかはたどり着くでしょうなｗｗｗ
向かっているわけですからなｗｗｗ違いますかな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32回～第34回の世界コンピュータ将棋選手権で強豪将棋AIチームを結構()なぎ倒してWCSC34第3位（5000えん）の快挙を達成し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NNUE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

おうちパソコン3990Xでどこまでやれるか見ものですなｗｗｗさすがにこのご時世では苦しいので重課金するかも知れませんなｗｗｗ

・定跡について

手作業で調整修正した居飛車定跡「1.21ジゴワット火葬砲フェルン姉弟子定跡」を使用しますぞｗｗｗ

もう何が何だかわかりませんなｗｗｗ

通称姉弟子定跡ですな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

もう後手番の勝ち筋は風前の灯ですなｗｗｗ強豪チームの先手番にはまったく太刀打ちできませんなｗｗｗありえない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青野流で必勝ですぞｗｗｗ

一手損角換わり：先手必勝ですなｗｗｗなぜか藤井聡太は苦手のようですな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指すといきなり評価値が終わりますな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NNUE評価関数のハイブリッド＋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

やはり定跡強化が勝利の鍵になりそうですなｗｗｗところで定跡は将棋AIなんですかなｗｗｗ

現時点では最強の氷〇彗dlsh〇gi連合が他をズタボロにする未来しか見えませんなｗｗｗ

・評価関数について

強いNNUE評価関数をベースに独自の学習で強化する予定ですなｗｗｗ

自分でﾅﾝﾄｶして強ければﾅﾝﾄｶするかも知れませんなｗｗｗnnue-pytorchｗｗｗ

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

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

ﾅﾝﾄｶ10ﾅﾝﾄｶｰ()、ﾅﾝﾄｶman()などを予定しておりますが現時点ではどうなるかさっぱりわかりませんなｗ

ｗｗ

・シードについて

今回は第3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枠は某やね〇さん枠がほぼ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

[English]

In the history of infectious diseases, a single drug has been easily resisted by mutations in the pathogen. In contrast, when multiple drugs are used simultaneously, each helps prevent the emergence of resistance to the others. Ryfamate derives from Rifamate which is the combination of two medicines for tuberculosis to retard the development of drug resistance. As with this method, Ryfamate aims to combine an NNUE alpha-beta search with deep learning (DL) Monte Carlo tree search to compensate for each other's shortcomings.

Ryfamate employs a modified majority voting system integrating a deep-learning-based evaluation engine and an NNUE-based evaluation engine [1]. By combining a deep-learning engine optimized primarily for GPU computation and an NNUE engine designed for CPU computation, Ryfamate effectively leverages the complementary strengths of both approaches. This hybrid configuration enables efficient utilization of computing resources within a single PC, significantly enhancing overall performance.

The author also proposes the Ryfamate Cross Network (RyfcNet) [2], an advanced deep-learning architecture specifically designed to enhance Shogi evaluation functions by effectively integrating the following distinctive components:

1. **S-Layer**: Conventional convolutions employing standard square-shaped kernels.
2. **C-Layer**: Extended convolutions utilizing kernels that span the full length of the input space across two or more dimensions, enabling selective dimensional shifts.
3. **A-Layer** / **F-Layer**: Spatial self-attention layer / channel-wise fully connected layers.

Standard deep-learning-based Shogi evaluation models [3] typically experience substantial computational challenges during training, particularly as the depth of the network increases. To address these limitations, RyfcNet introduces several strategic innovations:

1. Integration of multiple activation functions coupled with modified skip-connection architectures to alleviate gradient degradation and enhance feature propagation.
2. Implementation of ensemble learning and knowledge distillation techniques, leveraging multiple types of evaluation functions as teacher models to enrich learned representations.
3. Adoption of a hybrid optimization approach that selectively applies RAdam or LAMB optimizers alongside Stochastic Gradient Descent (SGD) at the layer level, significantly improving training efficiency and stability.

In recent years, large-scale neural models, exemplified by large language models (LLMs), have attracted considerable attention due to their exceptional representational power. Analogously, the innovations embodied within RyfcNet promise efficient scaling of Shogi evaluation models to accommodate greater parameter complexity, thereby improving predictive accuracy across diverse gameplay scenarios.

[Library]

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

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

* Ryfamate also uses a large amount of data that Kano-san, Yamaoka-san, Tayayan-san, nodchip-san have published.

[Reference]

[1] 水無瀬香澄, “Ryfamate 開発記と変則多数決合議,” コンピュータ将棋協会誌 Vol.36, 2025.

[2] Komafont, “Ryfamate Cross Network,” 第 33 回世界コンピュータ将棋選手権, 2023.

[3] 山岡忠夫, 加納邦彦, “強い将棋ソフトの創りかた Pythonで実装するディープラーニング将棋AI,” マイナビ出版, 2021.

Serenade アピール文書

谷合廣紀

2025 年 3 月 30 日

1　独自の工夫

- Stockfish の NNUE を参考にした、評価関数アーキテクチャと探索部
- 教師データのデータベース管理による、教師データの選別と定跡作成
- LLM を主体とした、複数エンジンのオーケストレーション

2　使用ライブラリ・使用データ

- ライブラリ: やねうら王, cshogi, stockfish
- 教師データ: nodchip/tanuki-.nnue-pytorch-2024-07-30.1, nodchip/shogi_hao_depth9

第３５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

あすねこ

- 開発内容：WCSC34の延長線上の開発
- 自己生成棋譜＝＞～118万棋譜(2025/03末時点, 千日手・持将棋を除く)
    - dl系：7000po ~ 24000po, NNUE系：3.0~9.0e6 nodes で生成
    - 30~150手を利用する場合、0.77憶局面(重複を除く)
    - GCTのhcpeファイルとマージして合計 2.7憶局面(重複を除く)を学習に利用
- rtx4090に最適化されたNetwork: AstraNET(自称)
- 教師データの学習時選別（新規）
- ReEvalによる追加学習（新規）
- USB計算式の変更（新規）
- 定跡作成：定跡の追加～合計50万局面(2025/04/E 時点)...（継ぎ足し）

WCSC35での開発内容

- WCSC34 で行った事を踏襲
- 色々なmodel 構造を試して学習可能性・速度・性能を評価した
  （基本的な理解を深める実験を行った）
- 学習時の工夫
  - 複数回の異なる乱数初期値での試行＆選別（学習開始時）
  - 学習時のネットワークのブロック順序の入れ替え（学習初期）
  - 異なる特徴を持つ学習データ・学習条件の交互使用（学習末期,lr が小さい事）
  - 学習時に教師データを評価し取捨選択
  - ReEvalによる追加学習
- 学習後の追加工(学習済のmodelの加重平均＆乱数による揺らぎの付加：後述)
- 探索部：UCB値計算式の変更
- 探索エンジンの切り替え(手数＆評価値)…(Ponderと両立できていないので使わないかも)

生成棋譜

- ベンチマークを兼ねた棋譜生成
  - NNUE vs. NNUE（5%位）
  - NNUE vs. DL（85%位）
  - DL vs. DL（10%位）
- DL系は 7000po~25000po程度で順次大きくしてきた
- NNUE系は 当初 3M nodes, 最近は 6~9M nodes
- NNUE系のエンジンは4種類利用

  (Hao,Tanuki-dr4,Daigorilla5,Suiden3)
  - dl側の勝率が55%~60%になる様に node数/po数を調整
  - 複数の対戦相手でベンチを行うのは重要
  - 特定の対戦相手のみで棋譜生成すると、指し手がその相手に似てくる

生成棋譜(cont.)

- Tanuki-dr4の棋譜が最も多い
- 複数AIの意見が分かれる局面では、Tanuki-dr4と同じ手を指す場合が多い
  - modelは、教師データの影響を強く受けている
  - 良いmodelを得るためには、良質な教師データを用意する事が重要では？
- 棋譜の質は重要！（数だけではない！）
- 指し手は教師データの影響を強く受けており、将棋の神様には ほど遠い
  （たどり来て未だ森林限界）

- **AstraNET(自称)のrtx4090への最適化（後述）**

- **教師データの学習時選択**
  - 学習中のmodelと意見の合わない学習データの削除(～5%)
  - どのようなデータを不採用とするかが難しい
  - 学習時間が概ね２倍になる
  - 効果は有ったがそれ程大きくはなかった=> ReEvalの方が良い

- **ReEvalによる追加学習**
  - 山岡氏のコード使用
  - 効果あり(劇的という程ではなかった)
    - 元になる model の規模と学習中のmodel の規模が大きくは違わないためではないか
    - 探索を行わない ReEvalの限界の可能性もある

定跡作成

- WCSC34 と同じ２段階作成(詳細はwcsc34のアピール文書で)
  - 先手が単調に勝った棋譜を沢山集める(R4400以上)
  - 最も多く選択された手は先手にとって悪い進行ではない可能性が高い
  - 2段階目でその進行が正しいか検証する（後手も同様）
- WCSC34 で使用した定跡に棋譜を追加
- 2025/04/末 時点で50万局面程度
- 後手番では相手により２種類の定跡を使い分ける予定

以下、詳細

1)AstraNET(自称)について

2)Modelの追加工について

3)スレッド生成並列化による高速化

4)UCB値計算式の変更

AstraNETの概要

- rtx4090の場合、kernel数256のResNETに対して同一の性能であれば70%程度高速
- 高性能GPUでの効果が大きい（効果は使用するGPUにより変わる）
    - 計算ユニットが多い(＝並列化の粒度が大きい)場合効果大
    - データ転送速度が速い（3090, 4090 & PCIE-4以上のMB) 場合効果大
- 適用する対象(将棋・囲碁等)にも依存する（次項）
- ブロック数が12ブロック以上で問題なく学習できる事を確認

  それ未満で学習できるかは未確認

- 2023年9月頃から採用

AstraNETの考え方(kernel数)

1) 過剰な表現力の削減

kernel数はデータ量に応じて最適な値がある

将棋のデータ量は9x9=81なので256だと大きすぎる（推測）

囲碁だと19x19(将棋の約４倍)なので256でも問題ないのでは（推測）

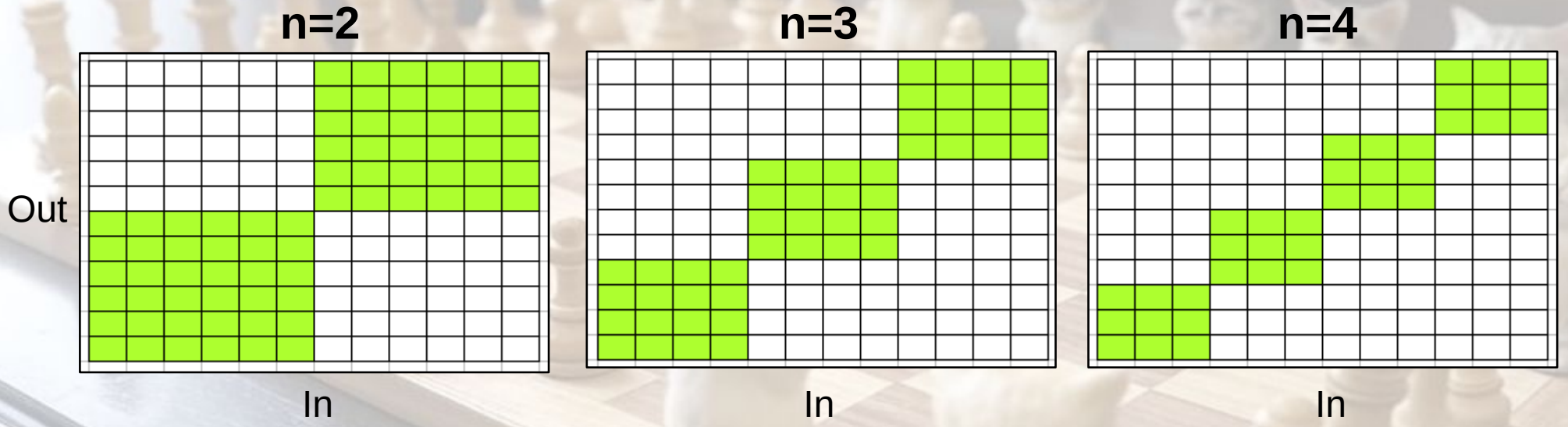
2) NVIDIAのGPUの並列化最小単位はGPU毎に異なる

使用するGPUで効率的に計算できるkernel数を採用する

rtx4090 では128が好適

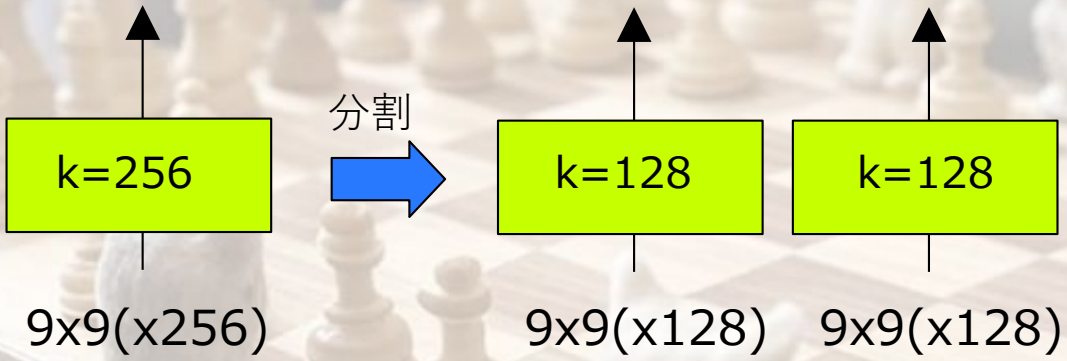
(1)と(2)の両方を満たすkernel数でネットワークを構成する

AstraNETの考え方(計算量の削減)



Conv2D()の対角部分(**緑色の部分**)のみを計算する事により計算量を削減する

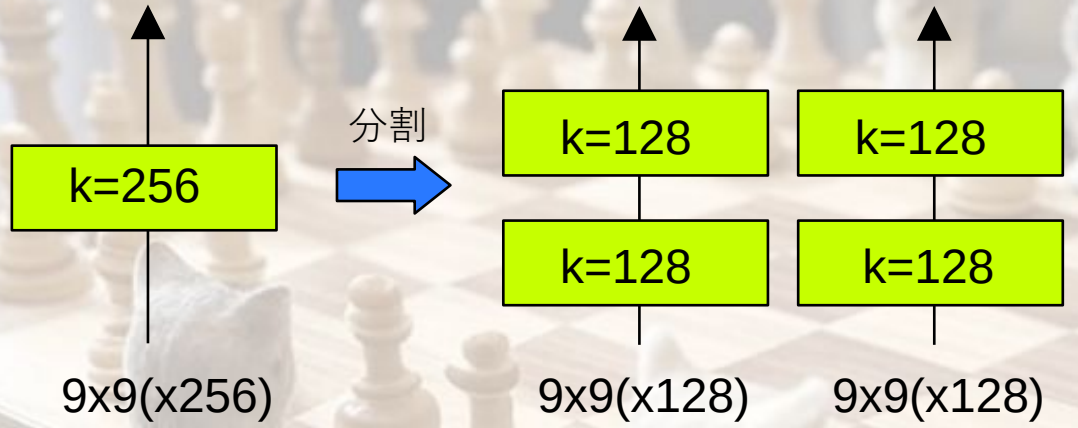
AstraNET(１)



データフロー：同じ
計算量：1/2
フィルター数：同じ
層数：同じ
非線形変換：1回(同じ)

大きなkernel数のConv2D()を、使用するGPUに最適なkernel数で構成されたResNETで表現する。RTX4090では k=128。

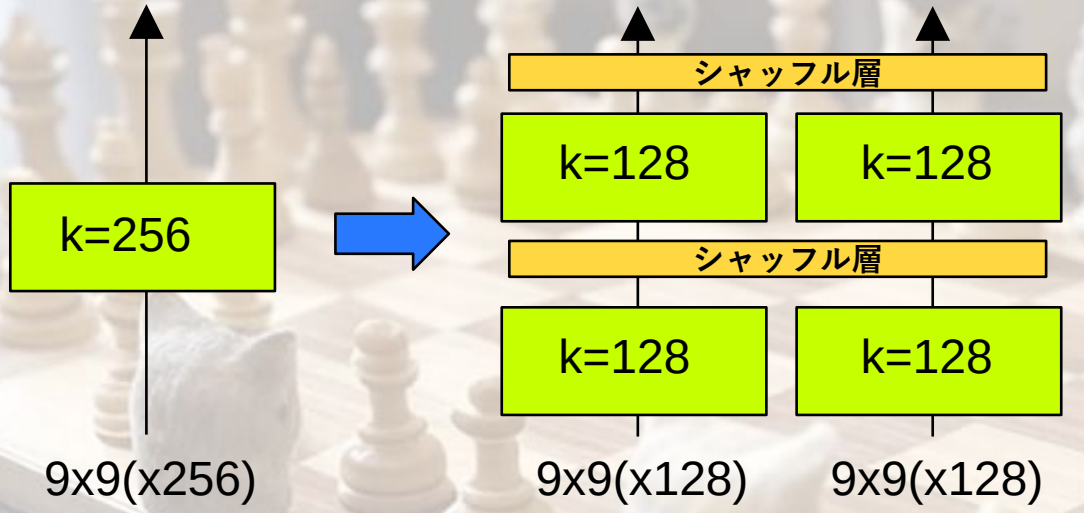
AstraNET(２)



データフロー：同じ
計算量：同じ
フィルター数：2倍
層数：２倍
非線形変換：2回(2倍)

・同じ計算量で、フィルター数を２倍、非線形変換の回数を２倍にする事ができ表現能力が増す
・層数が２倍になるので、元の3x3のフィルターよりも広範囲の5x5の情報を得ることができる
・ただしこの構造は、層数が多くなるにしたがって学習終盤にLossが十分に下がらないのではないかという懸念がある

AstraNET(３)



データフロー：同じ
計算量：同じ
(シャッフル層分を除く)
フィルター数：2倍
層数：２倍
非線形変換：2回(2倍)

必要に応じて シャッフル層を追加する(必ずしも毎層入れる必要はない)
シャッフルの仕方は、何通りか考えられる
(シャッフルの仕方によりスピード(nps)と必要GPUメモリ量が変わる)

シャッフル層のオーバーヘッドが発生するが、rtx4090の場合 k=256と比較して
同じ層数でほぼ同じ性能が出て1.7倍程度高速だった

AstraNET(cont.)

前項では n=2 の例を示したが、n=2,3,4,6 で上手く学習できる事を確認

a) (9x9,k=128x2,36b) : n=2　　...k=256 x 36 block 相当、前項の例
b) (9x9,k=128x3,24b) : n=3
c) (9x9,k=128x4,18b) : n=4
d) (9x9,k=128x6,12b) : n=6　　...k=768 x 12 block 相当
を試し学習できた

ただ n が大きくなるにしたがって学習が難しくなる様
強さは a)~d)で大きくは変わらなかった
　厳密には b), d) は少しnpsが低下しその分弱かった（ a と c は nps はほぼ同じ ）
　n が小さい場合 nvidiaのハードの制御機構に起因して 2 の冪乗以外で並列化の効率が落ちる

＊入力データ量が少ないと、大きすぎる n は逆効果になりそう
　dlshogi の入力データ量の場合 n=6 までは問題ない
＊GPUの種類により 最も効率よく計算できる k の値は変わると思われる(後述)

AstraNET(cont.)

- 今回使用する予定の model (V318)は
  (9,k=128x4,6b)+(9,k=128x3,4b)+(9,k=128x2,24b)
  と多段構造にした
- 入力側のデータを多く(k=512相当)し、計算量を増やさずに層数を増やすため
- 入力側は各ResNET毎に異なるデータを受け取り、総計で34blocksを持つ

SM数(左)と並列実行に好適なkernel数の関係

- RTX5090.....170　　k=170 (9x9の将棋には大きすぎる？)
- RTX4090.....128　　k=128 (256 は大きすぎる？)
- RTX4080_16GB…76　　k=76 (or 152 ?)
- RTX4080_12GB…60　　k=60 or 120
- RTX3090.....82　　k=82 (164 は大きすぎる？)
- RTX3080_10GB…68　　k=68 or 136

kernel 数がこれらの数より少し小さいのは問題ないが、少し大きい場合は計算の並列化への悪影響が大きいので避けるべき

類似例を見つけた...(2025/03/03)

- ShuffleNet，ResNeXt(Grouped Convolution) の考え方に近い
  - データをグループに分けて計算量を減らす
  - ShuffleNetと同じ理由でデータを混ぜる
  - 実装方法＆シャッフルの仕方は異なる
  - kernel数をGPUのSM数に限定して高速化するのはオリジナル

- 2023/09頃から採用して、wcsc34 もこの構造を採用した

追加工 --学習終了後の派生modelの生成--

1）勾配が緩やかになった時点で学習が終了する
2）そのため学習終了時には、Loss値の盆地の端にいる
3）盆地の中央方向に移動したい

・未知の局面でよりロバストになる事を期待

・盆地の底にはまだ緩い勾配が残っている可能性があるが、通常の学習では効率が悪い

・到達場所が極所解であれば、近傍により良い場所がある場合がある

・算術演算により良いmodelの周辺のmodelを作成し、ベンチマークでbetterな物であるか確認する
・より良い物が見つからなくなるまで繰り返す

Modelの追加工

- ２つ(複数)のmodelの加重平均により派生modelを作成する
  より良いmodelに成る可能性がある
  - ノイズ成分を減少させる効果
  - 学習終了時には「盆地」の端にいる可能性が高い
    中央方向に移動する可能性がある
- 学習済modelを定数倍する
  - 1.001倍~1.004倍が良い場合が多かった
    (結構センシティブ：FP8等は将棋AIでは無理？)
- model に乱数を付加する事により派生model を生成する

これらを実施する事により、学習終了時のmodelからR+50程度の向上が期待できる

Modelの追加工(cont.)

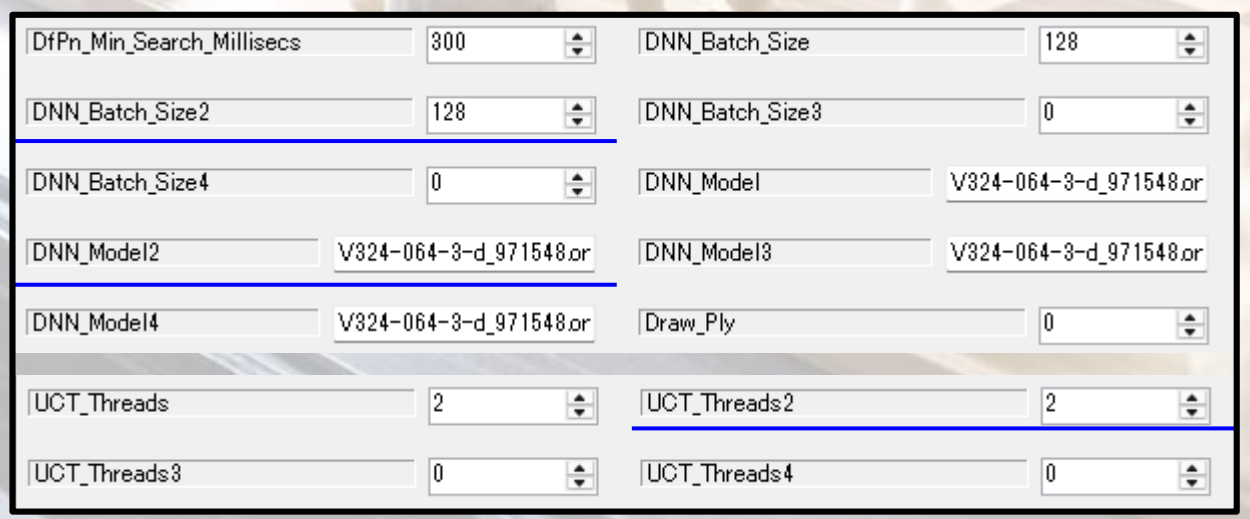
- 乱数は正規分布よりも一様分布がよかった
- 乱数の与え方は２通り試した
    - 読み出し単位で同じ乱数を使う
    - すべての数値に異なる乱数を使う
- 一様乱数の大きさのレンジは +/- 1.001 ~ +/- 1.004
  この数値を掛ける(学習の精度が良い場合は小さな数字が良い)
- 学習が十分に上手くいっている場合は、追加工による伸びしろは少ない

スレッド生成部の並列化による高速化

- スレッド生成部の並列化を採用した（wcsc34 でも採用した）
- 計算ユニットが多いGPUでは、計算ユニットがデータ転送待ちにならないような工夫が必要
- rtx3090/4090と今回検討したmodelの組み合わせでは効果あり
  - rtx4090 では30%高速化するmodelもあった
- 推論時のGPU使用率が90%以上にならない場合は効果がある可能性あり
- 効果の程度はHWのスペックに依存する（MBのPCIEの世代も重要）
- 効果がない場合もある

スレッド生成部の並列化(cont.)

- dlshogi では、GPUが 1 枚の場合は次の様な設定でスレッド生成部を並列化できる
- 効果は使用する GPU,MB と model のサイズにより異なる
- 特にGPU-MB間のバスの転送速度が重要(rtx3090, rtx4090でPCI-E4.0以上のMB必要)
- 概ねnpsの向上が10%以上であれば試す価値がある　それ以下では弱くなる可能性が大(*1)



図：設定の例 (128x2x2)

GPUが 1 枚のPCでは GPUIDは0となる
同時に GPUID=1を使う設定にしても
　GPUID=0として扱われる
この時 スレッド生成部が並列化される

*1: 本手法は標準的な探索に比べ探索の初期に探索ツリーの更新が遅れる
　そのため以降の探索に悪影響を及ぼす局面が存在するためと思われる

UCB値計算式の変更

- 次の一手問題を検討した時に問題によっては 実行毎に探索結果・nps が大きく異なる場合がある（結果が２値的に分かれる）
- 探索初期の乱数の出方に依存して、探索の集中する枝が変わるのが原因ではないかと推測 ＜＝これを対策したい
- 特定の条件下で、勝率項の重みが小さければ安定した探索ができるのではないか

- 一手１秒程度の探索で勝率の向上を確認
  (より長時間、例えば一手10秒思考での効果は未確認)
- ２番目の候補の探索回数が増えていた

UCB値計算式の変更(cont.)

- ucb_value = q + c * u * rate; を変形し
  ルート：　　ucb_value = f(q) + c * u * rate;
  ルート以外：　ucb_value = q + k * c * u * rate;　　k は定数
- f(q) は q の 2 次式、または 3 次式 ...比較した範囲では 2 次式が良かった
- 境界条件：f(0)=0, f(1)=1, 0 <= f’(1) < 1

- ７個の探索パラメータが先に調整されている事
- Temperatureの再調整が必要になるかもしれない
- 探索パラメータは７つあり調整が困難である事に対し、パラメータが１～２になるので調整が容易
  （探索パラメータの最適化が理想的にできているのであれば、この変更は必要ないかもしれない）

まとめ

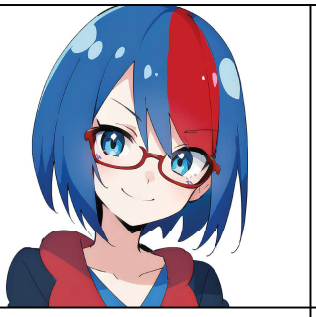
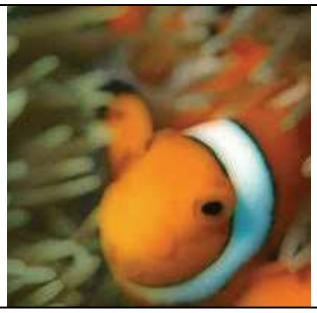
- 今回使用する予定の model (V318)は
  (9,k=128x4,6b)+(9,k=128x3,4b)+(9,k=128x2,24b)
  と、多段構造
- wcsc34 から R+120 程度 (@ rtx4090) 向上
  - 定跡有りで、floodgate基準で R4500位@rtx4090（推測）
- AstraNET、及び「追加工」の詳細を説明した
- スレッド生成部の並列化を採用した
- UCB値計算式を変更した
- 手数と評価値によるEngine の切り替えを実装した（詳細略:ページが足りない）

第35回世界コンピュータ将棋選手権

キーワードは「わかりやすさ」
大規模言語モデルを用いた
将棋初心者支援システムの提案

水匠チーム 創造研究開発部 熊田ゴウ(主任)/たややん/アイシア＝ソリッド/shimojolno

プロジェクトメンバー

			
熊田ゴウ	たややん	shimojolno	アイシア＝ソリッド
• リーダーとしてプロジェクト立上げから統括、開発を担う。 • 普段は主にユーザー支援のための研究開発に従事。 • VTuberでアマチュア四段。	• 水匠を強くする係＆DL系AIを用いて本プロジェクトに貢献する担当。 • 普段は弁護士！	• 先行研究調査を行い、本プロジェクトの位置づけを明確化。手法検討のサポートなど。 • 普段は企業のデータサイエンティスト。 • 3手詰めが精一杯。	• 将棋初心者を担当。最新のAI動向を共有しつつ、局面類似検索で爪痕を残す予定 • 普段はデータサイエンス VTuber として統計や AI の動画を投稿 • 棋力はウォーズ5級
X：@kumada_goh Youtube：@kumada_goh	X：@tayayan_ts YouTube：@tayayan_ts	X：@shimojolno reserchmap：asaya	X：@AIcia_Solid Youtube：@AIcia_Solid

研究サマリ

背景と課題

- 将棋人気が高まっているが、初心者向けの支援は不十分。
- AI評価値は初心者の理解を助ける一方、評価値以上の情報を与えられない。
- 自分で調べようと思っても名称不明で調べられない上、学習を支援する人員は不足しているため、初心者が取り残されている。

手法と結果

- 観戦者向けの理解支援：将棋知識DBとLLM(ChatGPT 4o)の活用で、人間に理解できる・納得感のある解説を生成。
- 初心者向けの学習支援：段位者AIと級位者AIを戦わせ、級位者によくある失敗と理由を解説。
- 多言語対応：上記システムは即座に多言語化可能。
- 結果、将棋初心者の将棋学習を支援し、継続的に将棋を楽しむためのシステムを構築した。

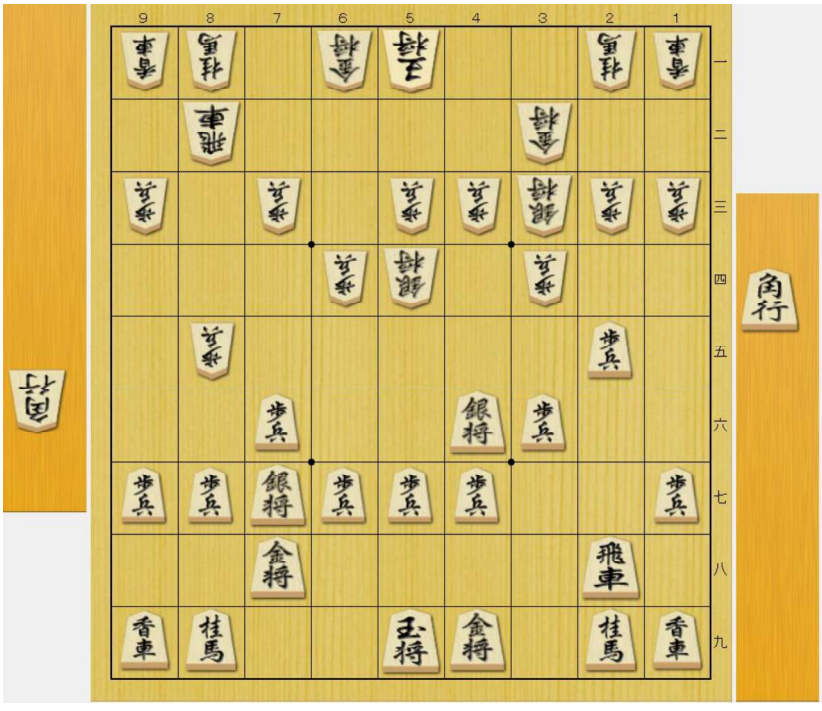
今後の展望

- プロの将棋の中継に対してもAIによる自動解説を行うなどし、広く将棋初心者に向けて価値を提供すること。
- システム内部の知識DBの改善やLLMのチューニング等により、プロが使えるようなシステムを開発すること。

抜粋

1.2.3 現代の将棋AIの限界：難しすぎる

ユーザー局面



システム出力

将棋AIによる出力：

評価値

-32

読み筋

▲６九玉(59) △４四歩(43) ▲５六歩(57) △５二金(61) ▲９六歩(97) △４二玉(51) ▲７九玉(69) △９四歩(93) ▲８八玉(79) △１四歩(13) ▲１六歩(17) △７四歩(73) ▲３五歩(36) △４三銀(54) ▲３四歩(35) △同　銀(43) ▲５五歩(56) △３一玉(42) ▲６六歩(67) △２二玉(31) ▲５八金(49) △６五歩(64) ▲同　歩(66) △３九角打 ▲３八飛(28) △８四角成(39) ▲６七金(58) …

抜粋

2.1 開発方針

将棋をわかりやすく。

問題

- 😇 難しすぎる
- 🤷‍♂️ 説明なし
- 😓 疑問が残る

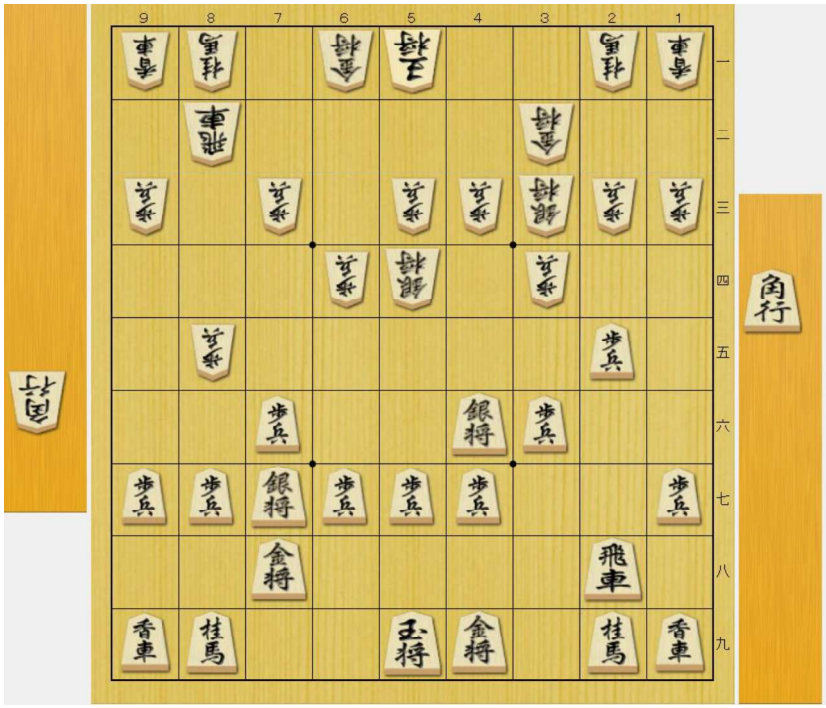
解決

- 🐣 初心者にもわかる！
- 💬 言葉でわかる！
- 🙋‍♀️ 疑問に寄り添う！

抜粋

3.1 開発結果1（手法A）

ユーザー局面



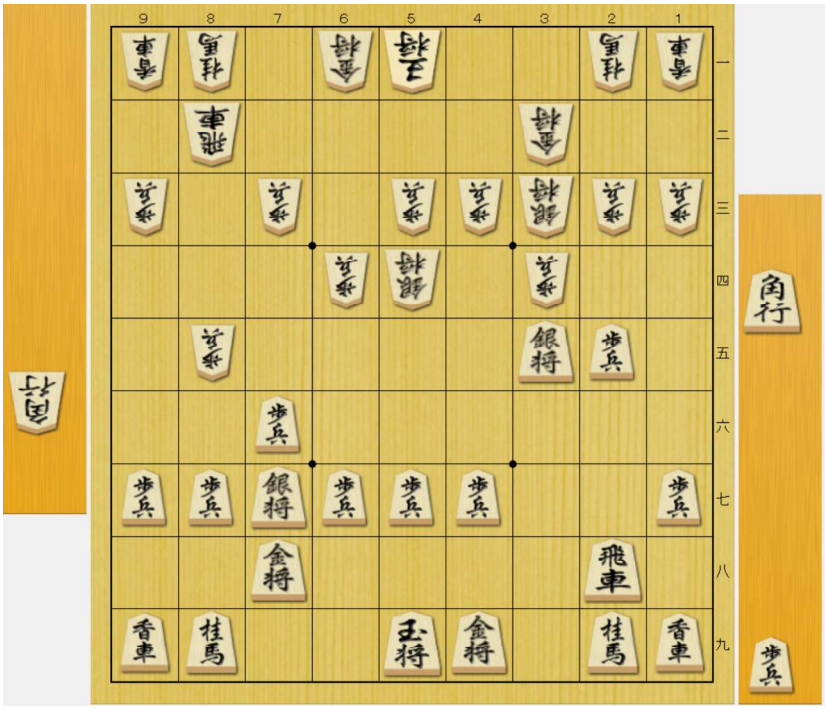
システム出力

知識DB × ChatGPTで
解説文を生成：
この局面は角換わり早繰り銀対腰掛け銀ですね。
このまま仕掛けると王手飛車になるので、玉を上がるか端歩を突いてください！

抜粋

3.2 開発結果2（手法B）

ユーザー局面



システム出力

級位者AIが王手飛車を喰らうその筋を解説し学びを提供：

この局面で、相手は△３五歩という手を狙っていると思われます。この手で、**銀を取ることを狙っているので注意しましょう。**

この局面で▲２四歩という手が見えるかもしれません。
しかし、▲２四歩、△同歩、▲同銀、△同銀、▲同飛、△１五角、▲６八玉、△２四角のように進めてしまうと、飛車を取られてしまうため、注意が必要です。

この局面では▲４六銀を指すことが考えられます。
▲４六銀、△４四歩、▲６九玉、△４五歩、▲３七銀という進行が一例で、相手の狙い（△３五歩）に対応することができます。

目次

1. 背景と目的

1.1 将棋ブームによる好機と課題

取り残された課題

- 藤井聡太八冠の誕生で将棋が注目を集めている一方、
  競技人口は減少傾向。*1

- スマホアプリの普及で「始める環境」は整ったが、
  「続けるための支援」が不足。

- **AI を活用した初心者支援**が拡充できれば、
  より**将棋界を盛り上げられる可能性**がある！

*1 公益財団法人 日本生産性本部 レジャー白書2024

1.2 初心者が抱える問題と解決できない構造

やり始めても続けられない

①初心者を待ち受ける将棋学習の壁

駒の動きや基礎の囲いを覚えた後、200超の未知の戦法が立ちはだかる。*1

将棋AIの普及は進んだものの、出力が高度すぎて初心者の学習には向かない。

将棋の難しさに魅力は感じるものの、学ぶための取っ掛かりが少なすぎる。

②初心者を支援する人が足りない。

人に教われば上記の悩みは解消するが、プロと普及指導員は約1200名。*2

将棋をやめてしまう理由に「指導者や教室がないこと」が約30%。*3

*1 本研究で列挙した結果。局面を見ても名前がわからず、勉強したくても調べることすらできない

*2 日本将棋連盟HPより

*3 文化庁 平成２９年度 生活文化等実態把握調査事業報告書 p33

1.2.1 指す将にとっての学習環境の例

AIにより学びやすくなっているが、まだ大きな壁がある。

- **アクティブな学びを得るのが難しい環境**

  棋書や詰将棋などでパッシブな学びはできるが、根気が必要で継続が難しい。
  対局〜検討等アクティブな学びができれば楽しいが、オンライン中心では難しい。

- **今のAIで学べること、学べないこと**

  評価値を見れば、悪手と最善手はわかる。
  しかし、それぞれの手の意味が分からなければ、やはり実戦には活かしづらい。

  ※例外的に、短手数の詰みが生じている局面で指し手の意味がわかることもある。
  しかし初心者にとっては、それ以外ほぼすべての局面で最善手の意味はわからない。

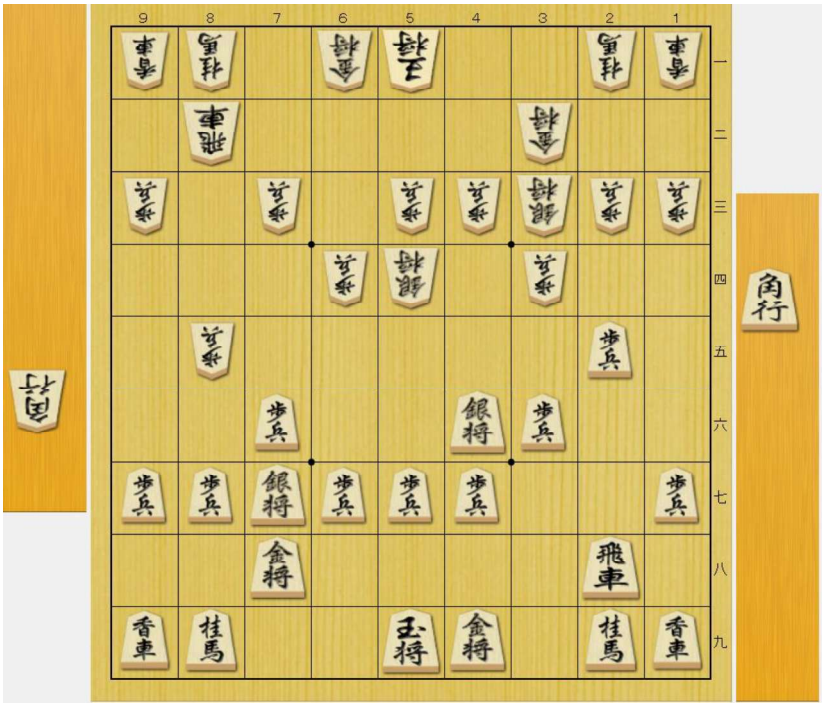
1.2.2 戦法の例：相居飛車の定跡

これでもまだ一部。多すぎる！

矢倉	急戦矢倉	角換わり	相掛かり	横歩取り	雁木	その他
3七桂4七銀 棒銀 早繰り銀 四手角 雀刺し 森下システム 脇システム	米長流急戦 阿久津流急戦 矢倉中飛車 右四間飛車 カニカニ銀 居角左美濃 4二飛戦法 3五歩早仕掛け	腰掛銀 58金型 木村定跡 48金29飛型 早繰り銀 棒銀 ４五桂戦法 右玉 後手一手損	原始棒銀 相掛かり棒銀 UFO銀 早繰り銀 腰掛銀 ひねり飛車 中原流 鎖鎌銀 塚田スペシャル	△3三角 △3三桂 △2三歩 △4五角 △8五飛 ▲3三飛成 相横歩取り 青野流 △４一玉戦法 勇気流 竹部スペシャル	ツノ銀雁木 旧型雁木	右玉 袖飛車 陽動振り飛車

1.2.3 現代の将棋AIの限界：難しすぎる

ユーザー局面



システム出力

将棋AIによる出力：

評価値

-32

読み筋

▲６九玉(59) △４四歩(43) ▲５六歩(57) △５二金(61) ▲９六歩(97) △４二玉(51) ▲７九玉(69) △９四歩(93) ▲８八玉(79) △１四歩(13) ▲１六歩(17) △７四歩(73) ▲３五歩(36) △４三銀(54) ▲３四歩(35) △同　銀(43) ▲５五歩(56) △３一玉(42) ▲６六歩(67) △２二玉(31) ▲５八金(49) △６五歩(64) ▲同　歩(66) △３九角打 ▲３八飛(28) △８四角成(39) ▲６七金(58) …

1.3 将棋AIの進化と課題

強さの追求と取り残される初心者

🤖将棋AIの発展	☹️既存の支援の限界
• 将棋AIは強すぎる！ • プロがAIの将棋を研究する時代 • 「初心者のためのAI」が未発達	• 既存の教材は一方通行 • 初心者が感じた実践的な疑問を解消できない • 人に聞こうにも教えられる人間が少ない

⇒ 初心者への新たな支援が必要！

1.4 研究の目的

初心者の「わからない」「不安」を「わかる」「楽しい」へ

1.人間らしいフィードバックやガイドで
初心者の疑問解消をサポートする

2.誰でも簡単に利用できるローコストな支援手法を探求

3.楽しみながら学び、続けられる環境を作る

1.5 局面解説AIに関する先行研究

局面の言語化を行ってきた研究たち。

	技術アプローチ	出力形式	評価方法	メリット	デメリット
亀甲ら (2017)	解説木を用いた手順の予測	局面の遷移を含む解説文	指し手予測モデルの精度評価	局面遷移を考慮	解説木が大きくなりすぎる
佐々木・関 (2023)	コーパス構築と機械学習モデルのFine-tuning	構成要素を分類した上での解説文	BLEU-2スコア	解説文の構成要素を体系的に分類	指し手の読み筋や戦型のみに特化
中村・小田 (2024)	局面価値と着手価値評価の数値データを言語化	囲碁特有の表現を用いた解説文	（未実施）	着手の特性（好手/悪手）の表現	評価実験が未実施
山内・河原 (2024)	手順のテキスト化と言語モデルのマルチタスク学習	対局の流れを考慮した解説文	ROUGE-L、BLEU、BERTScore	時系列の流れを考慮した入力形式	着手の意味理解が不十分

1.6 先行研究の課題と本研究のゴール

先行研究の課題：

解説文生成のための表現や言語モデルの選択がメインなため、

「初心者にとってのわかりやすさ」を十分に考慮した
実用的な支援システム構築には至っていない。

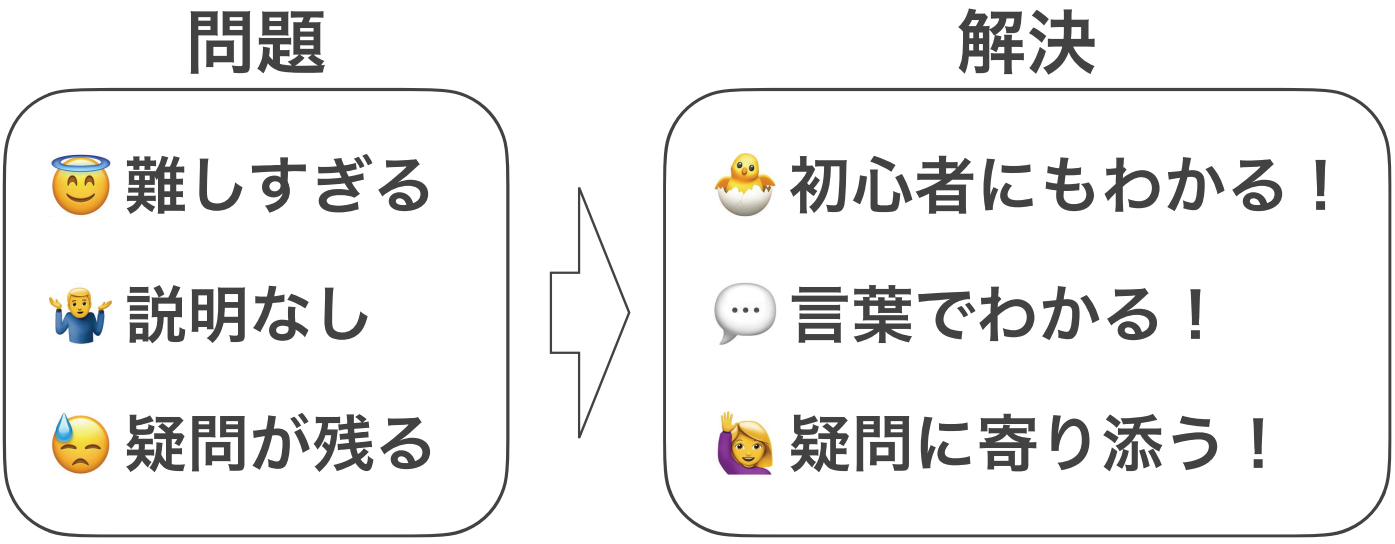
本研究のゴール：

局面の戦型認識と最適な次手推薦を100字程度の簡潔な自然言語で提示することで、初心者将棋への局面理解を促進する実用的な支援システムを構築すること。

2. 手法

2.1 開発方針

将棋をわかりやすく。



2.2 今回用いる2つの手法

将棋の勉強に重要な「知識」と「読み」を2つの手法でサポート

- **手法A：知識DBとLLMを活用して解説を生成する（知識のサポート）**

  将棋の局面（部分図）と、その狙いや注意点等の解説データを搭載したDBを作成。
  与えられた局面に類似する知識をDBから呼び出し、LLMを用いて解説を生成。
  ⇒ 現局面の意味合いや戦型の名前を示しつつ、今後の方針をわかりやすく解説する。

- **手法B：棋力差のあるAI同士を戦わせ、学びのある手筋と解説を生み出す（読みのサポート）**

  段位者と級位者の棋譜で追加学習し、それぞれ段位者・級位者の指し筋を再現したAIを作成。
  与えられた局面から段位者AIと級位者AIに対局させ、成り駒や駒の損得が生じる手筋を発見。
  ⇒ 級位者が陥りがちなミスを具体的な手順で示すことで、学びのある解説を生み出す。

2.3【手法A】人間の言語化の仕組み

有段者は局面をどのように言語化しているか。

①思い出す/読む

角換わり…
△1五角…
早繰り銀…

②言語化

「角換わり早繰り銀対腰掛け銀の形だね。」
「このまま仕掛けると王手飛車になるよ。」

⇒ AIを使って実装すればいい！

2.4【手法A】「わかりやすさ」を届ける仕組み

局面の取得	ユーザーが不明な局面で**ヒントボタン**を押す (その時点の局面情報を取得)
DBを検索	データベース上で**最もマッチする知識**が検索される
AIが言語化	マッチした知識を踏まえて、 **AIが言語化**する
画面に出力	マッチした知識と言語化された内容が **画面に出力**される

2.5【手法A】知識データベースと検索の仕組み

将棋の基本的な知識を蓄積、検索できる。

📚知識データベース	👩‍💻検索の仕組み
• 初心者向けの基本的な形とその形に関する説明を蓄積 • 戦法は200種、囲いは40種程度収録 (みんなで頑張って作成)	• ユーザーの局面図と知識DBの部分図を比較 ＝駒の並びが最も類似する知識を抽出 (完全一致じゃなくてOK)

2.6【手法A】知識検索の例：戦法の判定

ユーザーの現局面

マッチした知識(赤枠)と付随情報

```
Match 1:
Size: 9×5
Window:
[['k' '.' '.' 'n' 'l']
 ['.' '.' 'g' '.' '.']
 ['p' 'p' 's' 'p' 'p']
 ['s' '.' 'p' '.' '.']
 ['.' '.' '.' 'P' '.']
 ['.' 'S' 'P' '.' '.']
 ['P' 'P' '.' '.' 'P']
 ['.' '.' '.' 'R' '.']
 ['K' 'G' '.' 'N' 'L']]
Matched pattern: pattern_0
Similarity: 1.00

Description: 角換わり早繰り銀の形の例。このまま仕掛けると
△1五角で王手飛車になるので注意！玉を上がるか、端歩を突こう。
```

⇒ 情報を整形後、画面へ表示。

2.7【手法A】言語化の仕組み：大規模言語モデル(LLM)

人間らしく言語化してくれる。

- Open AI 社の「**ChatGPT 4o**」を利用した。
- 人間との会話が行える**言語処理AI**。
- ファインチューニング*などは行わず、
  公式のモデルをAPIで呼び出す形で利用した。

*特定の文章などを学習させ、専門的な出力を可能にする手法

2.8【手法A】言語化の仕組み：LLMへのプロンプト(命令)例

前提共有

付与情報

出力方針

ルール：
あなたは**将棋の局面図を言語化するスペシャリスト**です。
①の情報を踏まえ、②の方針に従って出力してください。

①あなたに与えられる情報は以下のとおりです。
- 現在の局面図:
- 先手視点の検索結果:
- 後手視点の検索結果:

②以下のルールに従って局面についての方針を示してください。
- **簡潔な日本語で**ユーザーに指針を示す
- **将棋に詳しくない人でもわかるように**話す
- **最大100字程度で**出力する

2.9 【手法B】庶民的な棋力のAIの作り方

級位者・段位者のデータでFine-Tuning（微調整）

①まず将棋AIを作ります

③級位者っぽい手を指す将棋AIが完成！

②級位者の棋譜データを与え、級位者の着手を再現するよう学習

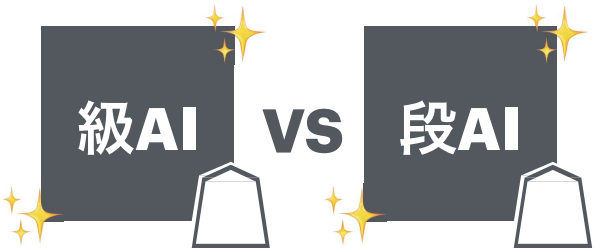
※一連の流れは、近年のLLMの学習方法と類似
まず一般的なタスクを学習させ (Pre-Training)、その後個別のタスクに調整 (Fine-Tuning)
今回の場合、まず将棋AIを作って将棋を理解させた後、級位者の考え方を学習させるイメージ

2.10【手法B】庶民的AIを用いた解説文生成

級位者 vs 段位者を通して庶民的でわかりやすい学びを得る

- **以下のステップで指し手と解説文を生成**

1. 指定局面から**級位者AIと段位者AI**に対局させる
2. 段位者AIが**駒得**や**成り駒作り等**に成功したら終了
3. 着手の推定選択率*を加味しながら解説文を生成



- **狙いと得られる効果**

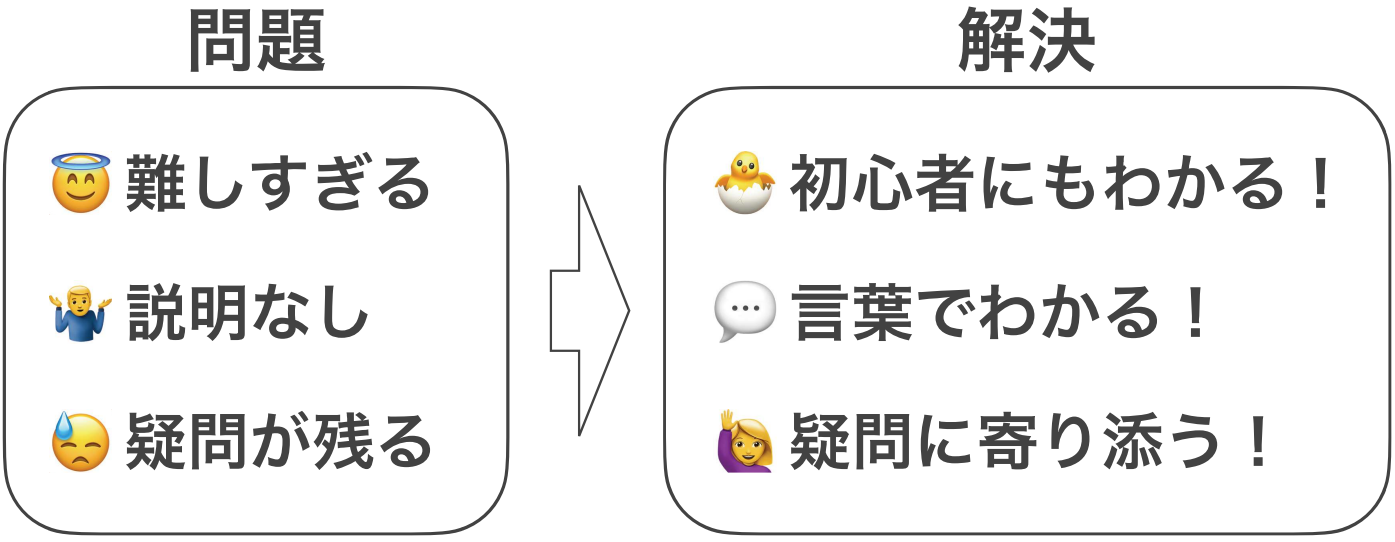
級位者AIが**初心者にありがちなミスを発見**し、段位者AIが**人間に理解可能な手筋で咎める。**
これにより、初心者でも理解可能かつ、深い学びにつながる解説を提示できる。
また、推定選択率を絡めることで「どの程度自分でも指せないといけなかったか」を
分かりながら振り返りを行える

*Policyという言葉をたややん氏が日本語化したもの。AIによる、次の一手でどの手を選ぶべきかを示す確率分布のこと。

3. 結果

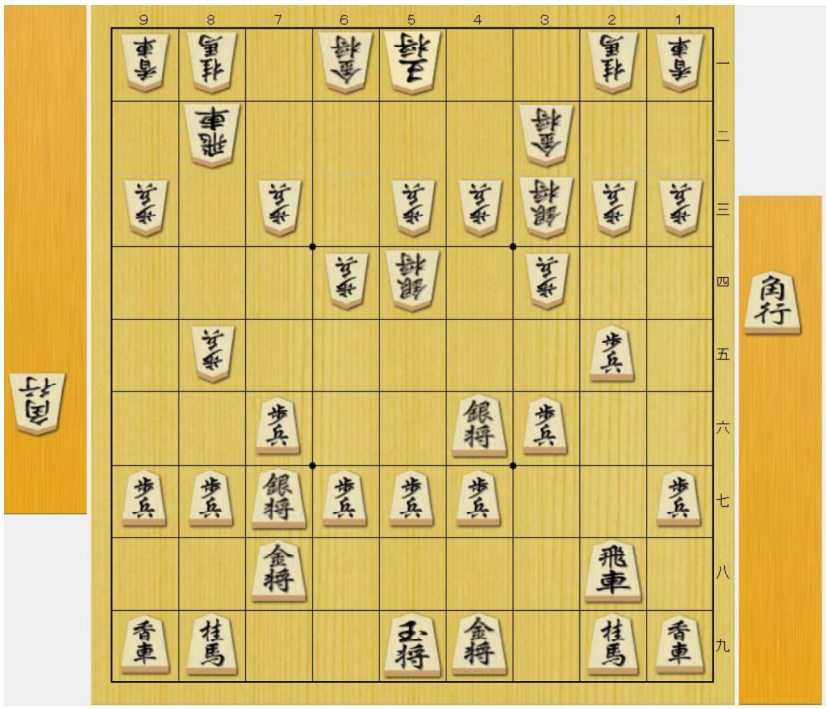
2.1 開発方針（再掲）

将棋をわかりやすく。



3.1 開発結果1（手法A）

ユーザー局面



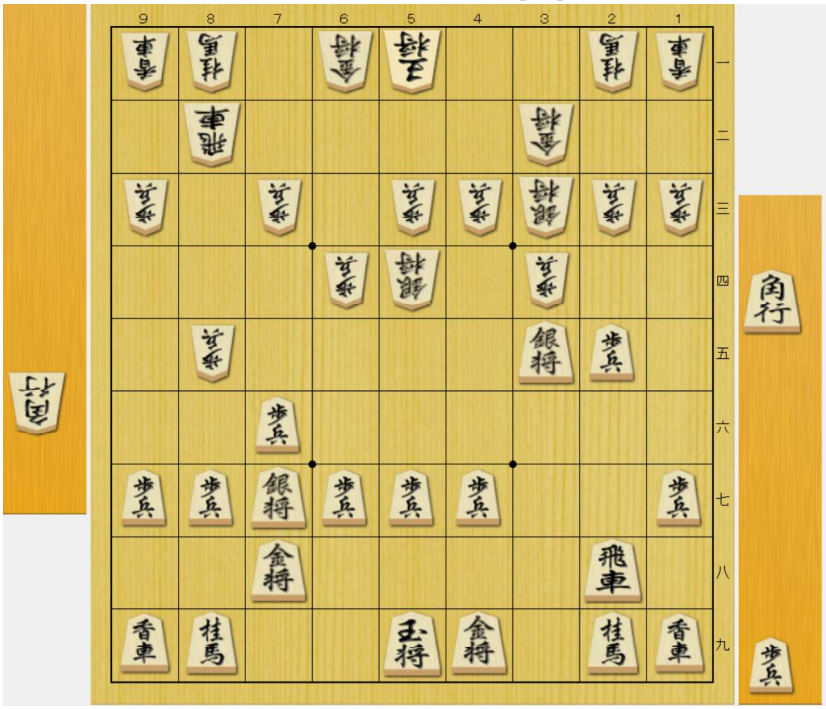
システム出力

知識DB × ChatGPTで解説文を生成：

この局面は**角換わり早繰り銀対腰掛け銀**ですね。
このまま仕掛けると**王手飛車になる**ので、**玉を上がるか端歩を突いて**ください！

3.2 開発結果2（手法B）

ユーザー局面



システム出力

級位者AIが王手飛車を喰らうその筋を解説し学びを提供：

この局面で、相手は△３五歩という手を狙っていると思われます。この手で、**銀を取ることを狙っているので注意しましょう。**

この局面で▲２四歩という手が見えるかもしれません。
しかし、▲２四歩、△同歩、▲同銀、△同銀、▲同飛、△１五角、▲６八玉、△２四角のように進めてしまうと、飛車を取られてしまうため、注意が必要です。

この局面では▲４六銀を指すことが考えられます。**▲４六銀、△４四歩、▲６九玉、△４五歩、▲３七銀という進行が一例**で、相手の狙い（△３五歩）に対応することができます。

3.3 開発結果：将棋がさらにわかりやすくなった。

まるで人間に指導してもらえているような体験

- **初心者にもわかりやすい**
  手順や評価値だけでなく**やさしい言葉で説明する**ため、**初心者の疑問を解消！**
- **指導してもらえているような体験**
  言葉によるガイドにより、**人間に教えてもらっているような安心感**を得られる！
  知識がなくても次の一手を考えやすく、**ストレスなく純粋に将棋を楽しめる！**
- **離脱防止&継続促進**
  指した将棋に関連する知識が自動で提供されるため、**効率的に学習を行える！**
  「わからない→やめる」ではなく、「わかる→続けたくなる」

3.3.1 AI活用例①：ぼこぼこに負けて途方に暮れたとき

次こそは勝てるように。

- **初心者はまず序盤で劣勢になり、悔しい思いをする。**
  「なぜか駒を全部取られた……」「よくわからない何かが起きて負けた……」
  なので学んで対策し、次は勝ちたい。
- **しかし、形の名前がわからないため検索しようがなく、学べない。**
  そもそも、**「定跡を知らなかったから負けたのか？」「ただ力負けしたのか？」**
  という疑問が残り、解消されない。

⇒AIのサポートで、悔しい時すぐに身のある学習ができる！

3.3.2 AI活用例②：対局相手の事前対策をしたいとき

本番で力を出し切れるように。

- **対局相手が事前にわかっていて、相手の棋譜を持っている場合もある。**
  相手の棋譜からは、相手の得意としている戦法を学べるはず。
  さらには、相手が負けている棋譜からは、**苦手な戦法を学べるはず**である。
- **万全の状態で迎えたいのに、対策できないもどかしさ。**
  しかし、**相手の棋譜を見ても、居飛車か振り飛車かくらいしか分からない。**
  なので、対策もできない。（←本当にできない。なかなかもどかしい）

⇒AIのサポートで、事前にしっかり対策できる！

3.3.3 AI活用例③：プロの将棋がわからないとき

解説をお願いします。

- **プロの将棋は難しい。**
  そもそも将棋が難しいのに、プロはさらに複雑なことをやっている(らしい)
  解説してくれるプロがいるときは、理解できて楽しい。
- **解説してもらえないと何もわからない。**
  しかし、中継の間ずっと解説してくれるわけではない。
  解説がない間は静かに盤面だけが映されていて、**初心者は取り残される。**

⇒AIが基礎知識や手順を解説し、安心して楽しめる！

4. 今後の展望

4.1 観る将にとっての将棋AI

「難しいけど、観ていてとても楽しい」(級位者代表：アイシア)

- **早指し戦や最終盤は今のAIだけでもかなり楽しい。**

  指し手も早いし、評価値が大きく動く。**「最善手以外悪手」だと手に汗握って観戦できる。**

  （第91期ヒューリック杯棋聖戦五番勝負第1局藤井竜王vs渡辺棋聖（当時）の連続王手など）

  現代の将棋観戦は、数字の大小を見るだけでも楽しめる（超簡単！）のに、

  奥に底知れぬ深みと人間ドラマがあり、歴史もある。

  おそらく、**観戦される娯楽の中で最も良い特性を持っている！**

- **しかし、序盤中盤の観戦は楽しみにくい。**

  初心者にとっては、AIの示す手の意味はわからないし、評価値の違いもわずか。

  プロの解説や、勝負飯等の話題がないと、観続けるのは難しい。

4.2 広く将棋を普及するために

- **西尾明七段（将棋連盟常務理事）「将棋ChatGPTの開発が進めば人間の学習機能向上だけでなく、さまざまな将棋サービスの向上が期待できる」** *1
- **もし、プロの対局中継に本システムを導入したら……**

  例1：プロの解説がない時間に、システムが解説。

  ⇒視聴者の理解を助け、将棋をもっと学べて、もっと楽しめる！

  例2：中継記者に本システムを活用いただき、業務時間を短縮。

  ⇒人間にしかできない取材という仕事が、さらに高品質になる！

  例3：LLMによる柔軟な多言語対応で、誰にでも「わかりやすく」。

  ⇒生まれた場所に関係なく、将棋を楽しめる！

*1 “「なぜその手を選んだか」　将棋AIが教える時代到来へ” 2024.03.25 NIKKEI BizGate

4.3 さらに高度なシステムについて

- **もし、プロのために本システムを改善したら……**

  今回は最も多くのユーザーに価値を提供できるよう、対象を初心者とし、基礎的な出力のみを目標とした。

  今後は、システム内部のデータを精細化、LLMのチューニング等、探索システムの開発等を行うことで、プロにとっても有用な出力を行える可能性がある。

- **プロに焦点を当てたシステム開発。**

  プロにとって当たり前の知識、当たり前の言葉遣い、第一感。
  それらの模倣や、強いAIとの組み合わせなど、研究の余地がある。

5. 引用文献

引用文献

[1] 亀甲博貴, 森信介, 鶴岡慶雅: "将棋解説文生成のための解説すべき手順の予測", 情報処理学会論文誌, Vol.58, No.12, pp.2070-2079 (2017).

[2] 公益財団法人 日本生産性本部 レジャー白書 (2024)

[3] 佐々木謙人, 関洋平: "将棋解説文の構成要素を考慮した解説文生成手法の検討", DEIM Forum 2023, 1a-7-5 (2023).

[4] 中村貞吾, 小田直輝: "着手価値評価を考慮した囲碁棋譜からの解説文生成", 2024年度電気・情報関係学会九州支部連合大会, 09-1P-04 (2024).

[5] “「なぜその手を選んだか」　将棋AIが教える時代到来へ” 2024.03.25 NIKKEI BizGate

[6]文化庁 平成２９年度 生活文化等実態把握調査事業報告書

[7] 山内悠輔, 河原大輔: "手順のテキスト化による将棋解説文生成", 言語処理学会第30回年次大会発表論文集, pp.3197-3202 (2024).

6. 今回の水匠の工夫

今回の水匠について

- **将棋初心者支援システム以外の部分（水匠の棋力の改善点）**
  NNUE（HalfKP512x2_8_64）を使用。
  nodchip氏の工夫（DL水匠で教師データの評価値を付け直して学習させる）を採用。
  参考：https://nodchip.hatenablog.com/entry/2024/12/25/000000

- **上記工夫に加えて...**
  Lambda = 1.0（勝敗項を考慮しない）で学習することで棋力向上。
  局面データをuniqueなものとする（重複を排除する）ことで棋力向上。
  Ryfamateモデル（感謝：Komafont氏）を活用することで棋力向上。

7. 謝辞

謝辞

知識DB作成に尽力してくださった皆さん。どうもありがとう！！

写真	名前	X	Youtube
	美野辺沙羅@怨霊VTuber	@Sara_Minobe	怪奇放送局野辺チャンネル
	甘党あずを🍵🍡Vtuber	@amatou_azuwo	Azuwo Ch. 甘党あずを
	一梨透🍮将棋系VTuber	@ichilitre_V	一梨透
	⛩鷺宮ローラン【プロe棋士Vtuber】プロゲーマー👽	@SaginomiyaL	鷺宮ローラン（バーチャル将棋星人Vtuber）
	五反田えぬ	@Gotanda_N	五反田えぬ
	四宮式@小説系VTuber	@YotsumiyaS	四宮 式 -Yotsumiya Shiki-
	篠崎マコト	@thekeymaple_1	篠崎マコト

WCSC35 godflygoooon アピール文章

今回は定跡を中心かつメインに生成した。先手番定跡に引っかからない対策を考え尽力を尽くした。
工夫した点をファイル別に記載する。
▲定跡
まず先手番としては初手▲26歩をほぼ固定させ枝を掘りやすいように基盤を立てたファイルを基本定跡とした。後、たややんさんの公開したプログラムを利用してふかうら王（dlshogiベース）を使用して4GPU（RTXミドルクラス）マシーンにて一局面１０万ノードで約２０万局面相当を追加した。
プログラムがバグっているのか定跡データベス形式で出力させるとファイルとして読み込めないことがあるので、sfen形式で出力させ、指し手の確立を手動で変更した。微調整しながAIに指し継いでもらい、出現確率の高い順から確率的に指すようにした。
後手番の秘策としては秘密としておく。

△評価関数
あまり変更点はなく、電竜戦のバージョンからは＋100ほどしか強くなっていない。Suisho10beta2と検証を行い、現在の精度は指定局面、20手目から対戦させ1000戦　6割程度だ。
追加学習として、Suishoで生成したデータを10億程度（dlshogiにて評価値を変更したデータも半数含む）とアピールポイントとして、ふかうら王で生成した棋譜約30万（NNUE VS dl　、dl15b VS dl20bなど）を最新の水匠で深さ10にて評価を付けた教師データも利用した。
はっきり下部の方は効果があるかわからない。

▲探索部
変更は特にないがClnag21向けにビルドした。

利用したライブラリ
ふかうら王　水匠　水匠15b やねうら王

AobaZero の 2025 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]。2025 年 3 月 31 日現在、7109 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 +0 ELO

追試終了後からは +260 ELO、去年の選手権からは +0 ELO で、ほぼ同じです。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

昨年は振り飛車のゼロからの強化学習「Aoba 振り飛車」[*3]を主に開発していたので目立った進展はありません。ただ、生成された振り飛車の棋譜を AobaZero の棋譜に 20 %から 25 %ほど混ぜると 128x10 block の小さいネットワークだと +50 ELO ほど強くなりました[*4]。

AobaZero は振り飛車の棋譜が 2 %しか含まれていないので、もう少し多様な戦法を学習させた方が強くなるのかもしれません。ただ 256x20 block の通常のサイズのネットワークだと 25 %混ぜてもほぼ同等に強さでした。

5　対振りと宣言勝ち対策をした dlshogi 互換モデルで出場予定

選手権では dlshogi の互換モデルで出る予定です。ネットワークのサイズは 384x30b。NN の入力には手番、手数を追加する予定です。手数は 320 手を 8 分割して 40 手ごとに全面 1 にします (手数で 8 面)。学習棋譜は Aoba 振り飛車の棋譜を 25 %混ぜます。これで相手が振った時に極端に勝率が上がるのを防げると思います。また持将棋で敵陣に駒が入るように 400 手以降で敵玉に入ってる枚数が 0 枚なら勝率を 0.50 に、9 枚で 0.95 に、する補正をしています。300 手と 250 手以上も勝率が上がりすぎないように小さく補正を。探索勝率を無理やり下げて、dlshogi の学習で勝敗結果と混ぜる割合を 0.333 から 0.500 にしています。あと時間があれば KataGo の NN 構造を試してみるつもりです。

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 300 万局面ほど探索させて作成する予定です。

7　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。

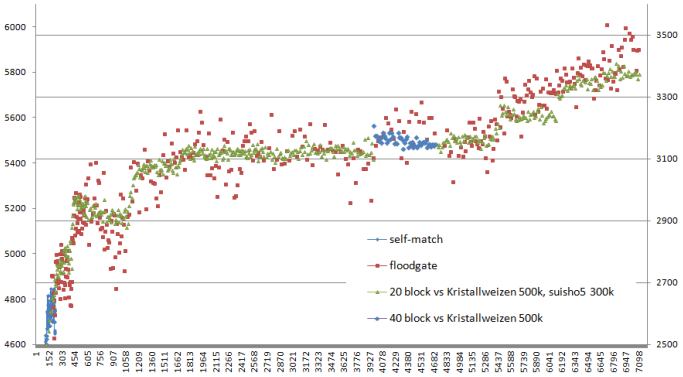


図 1　棋力の推移。右軸が floodgate のレート、横軸は棋譜数 (万)

8　今後

選手権後は 320x30b に移行して学習開始局面集を利用して多様性のある棋譜を作ってみようと考えています。

[*1] https://github.com/kobanium/aobazero
[*2] https://github.com/kobanium/aobazero/issues/54
[*3] http://www.yss-aya.com/furibisha/
[*4] http://www.yss-aya.com/bbs/patio.cgi?read=158&ukey=0

9　6 年で 7100 万棋譜

7100 万棋譜、という膨大な棋譜を 6 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。

ponkotsuアピール文書

竹内元気

目次

- 将棋AI ponkotsuの2024年での採用手法
- 2025年の改善手法

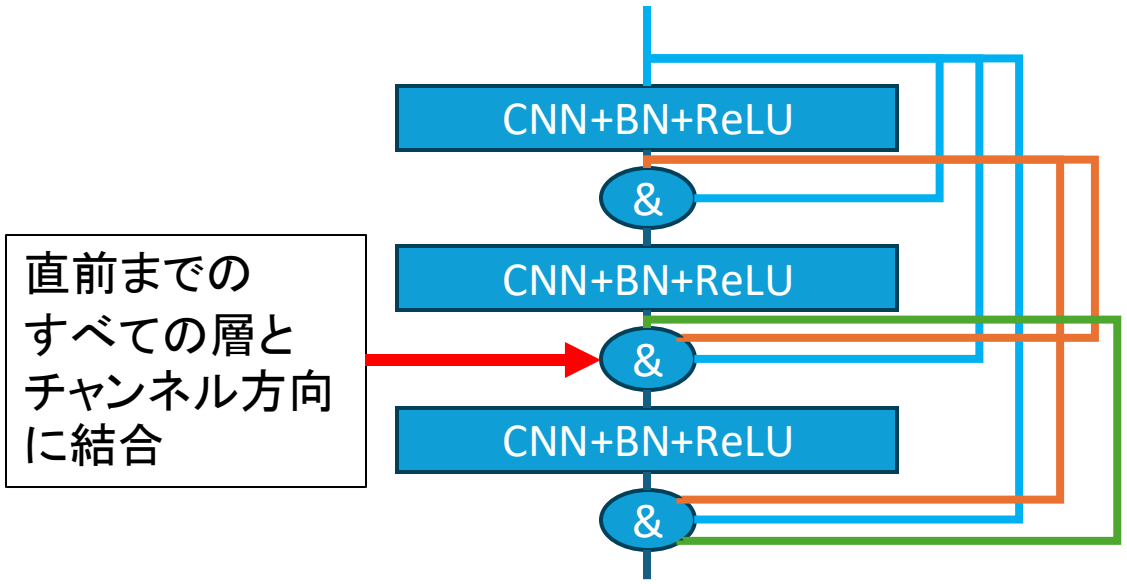
目次

- 将棋AI ponkotsuの2024年での採用手法
- 2025年の改善手法

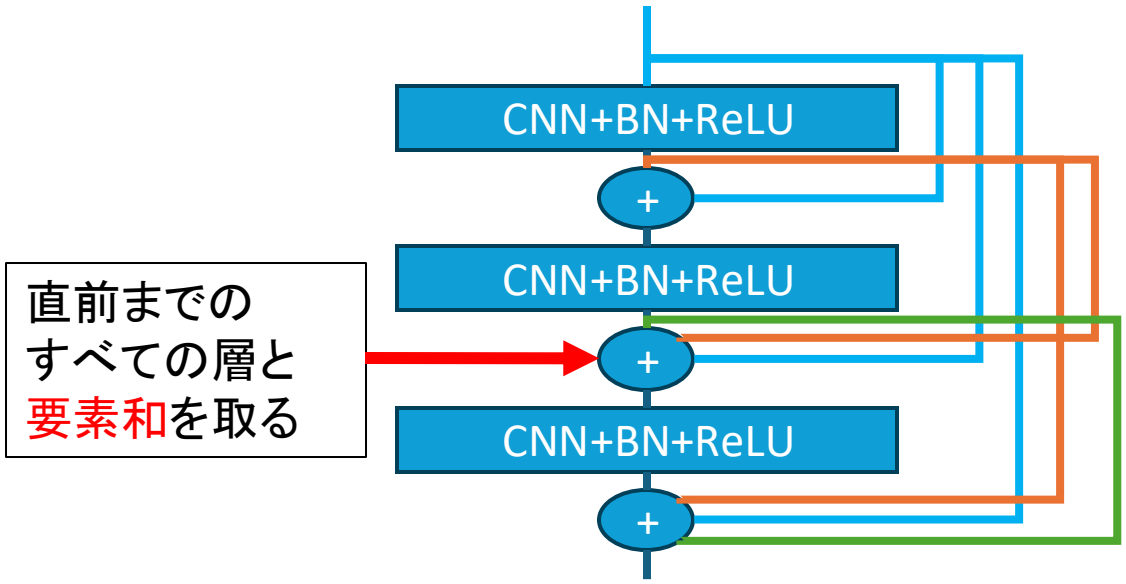
ネットワーク構造の改良

- 課題: 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
  →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

学習率スケジューラの変更

- 課題: 手動によるパラメータ設定でスケジューラを設定する必要がある
  →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

結果(識別精度)

- モデル構造
  - ponkotsu-2023: 従来版DenseNet、10ブロック
  - ponkotsu-2024: 改良版DenseNet、15ブロック
- epoch数
  - ponkotsu-2023: 365 epoch
  - ponkotsu-2024: 270 epoch
- テストデータ(floodgateの2017年～2018年の棋譜)による Policy(次に指す手)とValue(勝者)の精度
- Policy・Valueともに2024年バージョンのほうが高精度

Model	Policy Accuracy(%)	Value Accuracy(%)
ponkotsu-2023	45.5	74.2
ponkotsu-2024	**51.2**	**77.0**

目次

- 将棋AI ponkotsuの2024年での採用手法
- 2025年の改善手法

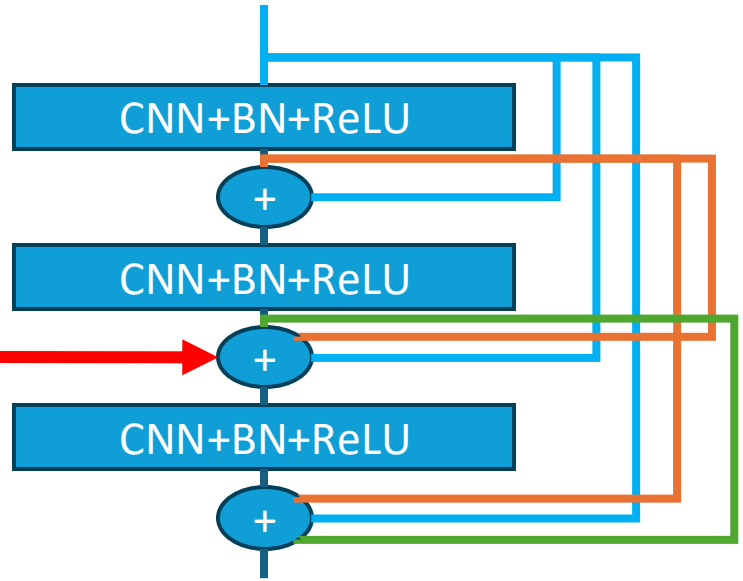
2024年からの改良内容

- ネットワーク構造の改良
- 学習データのアップデート

ネットワーク構造の改良

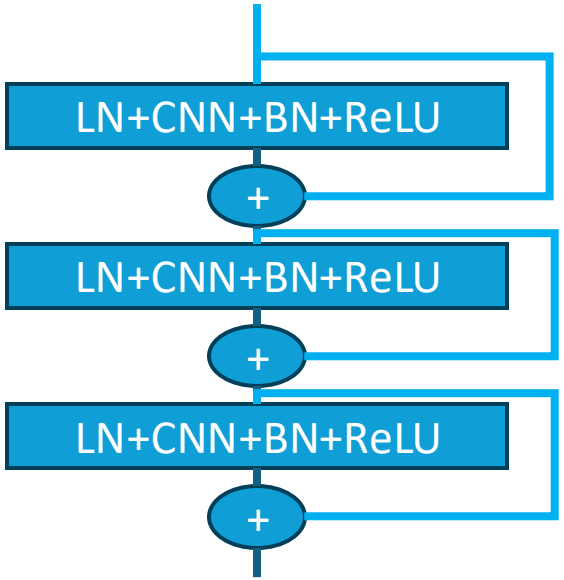
- モデル構造を改良DenseNetからResNetに変更
  - ResNetにLayer Normalizationを追加
    - 事前層正規化を実施

2024年版



直前までの
すべての層と
要素和を取る

2025年版



※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

事前学習データ

- 事前学習データを以下に変更

種類	データの内容
shogi_hao_depth9	将棋AI Haoを使用し探索深さ9で自己対局したデータ(リンク)
shogi_suisho5_depth9_entering_king	水匠5を使用し探索深さ9で自己対局した対局のうち、入玉将棋になった対局の棋譜を集めたデータ(リンク)
AobaZeroの棋譜	将棋AI AobaZeroを使用しプレイアウト数1600または3200で自己対局したデータ(リンク)
tanuki-.nnue-pytorch-2024-07-30.1	tanuki-.nnue-pytorch-2024-07-30.1を使用し探索深さ9で自己対局したデータ

学習データ

- 2024年に使用したデータのうちfloodgateの棋譜をアップデート

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～2024年3月のレーティング3800以上のプレイヤーの棋譜	floodgateの2019年～2025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	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)	アップデートなし
GCTの棋譜	2021年に行われた世界コンピュータ将棋選手権で14位のソフトの強化学習で生成した棋譜	アップデートなし

参考文献

- ReduceLROnPlateau — PyTorch 2.2 documentation
(2024年3月27日閲覧)

名人コブラアピール文書 (WCSC35)

概要

2025年、名人コブラのテーマはディープ浪漫！ 基本的にdlshogiを利用させてもらっていますが、ディープ浪漫をさらに向上させます。

改良点

- ディープは深い！
    - 深さこそ正義！ 昨年の本家dlshogiが30ブロックなので、名人コブラは更に深く、40ブロックに増やします。（でもフィルター数は減らします。）
- ディープは特徴量を加工しない！
    - 入力データを加工して使うのはディープ浪漫に反します。局面データをそのまま食らうのがディープ浪漫！ 駒の利き数や入玉の点数は甘えです。

使用ライブラリ

- dlshogi
- 水匠教師データ

以上

- **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当初は白ビールの強化学習データを流用するなどして，最高精度を目指す大きな深層学習モデルを作成しようとする試みである．探索無しでプロ棋士レベルの強さに到達したため当初の目標は達成したと考えている．

2022 年度は幸いにも準優勝という好結果を得たがその後モデルサイズの拡大には GPU メモリの壁があるためハードウェア的な制限で頭打ちの段階である．そもそも強化学習のステップにおいても 2020 年の段階で 1 ステップ 2 カ月規模に達し，最終的に 1 年半程度の規模となっている。それでも教師データ数は数十億局面程度であり，大規模学習へのリソース不足が明白である．

モデル単体の強化とは別に強化学習のアルゴリズムレベルの改造を進めている．小規模実験レベルでは AlphaZero などの欠点を克服する可能性を示しているが，高負荷実験での性能は未計測である．例年似たような表現となるが間に合えば新型，間に合わなかったら旧型で参加することになる予定である．

また，ハードウェアとしては電竜戦第 3 回ハードウェア統一戦で使用した統一ハードウェアを用いることで実機のパフォーマンスを示すことができたらと考えている．

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

WCSC35 nshogi アピール文章

中屋敷太一

2025/03/16

概要

- 基本的に AlphaZero[1] の再現実装
  実装は OSS として公開
  将棋のルール部分 https://github.com/nyashiki/nshogi
  思考部分 https://github.com/nyashiki/nshogi-engine/
- 強化学習に Gumbel AlphaZero[2] の棋譜生成アルゴリズムを使用
  ランダムな重みから学習
- ニューラルネットワークの構造は 30×320 の ResNet

[1] David Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. In: Science 362.6419 (2018).

[2] Ivo Danihelka et al. Policy improvement by planning with Gumbel. In: International Conference on Learning Representations. 2022.

工夫点

強化学習での工夫点

- 様々な最大手数設定（160 手から 1024 手）での自己対局
  - 勝勢のときに早く勝つ手（延命されない手），および
  - 敗勢のときに延命する手を学習できることを期待
- 引き分け点数設定による自己対局データの勝率調整
  - 先手勝ちが多いときは，引き分けを後手有利に
  - 後手勝ちが多いときは，引き分けを先手有利に
- MCTS の simulation 回数の手数での調整
  - 序盤，終盤は少なめの simulation 回数 (64 から 256 程度)
  - 中盤は多めの simulation 回数 (128 から 512 程度)

学習期間と棋力

学習期間

- NVIDIA GeForce RTX 4080 を約 10 ヶ月使用
  約 7500 万棋譜を生成

棋力

2025 年 3 月 16 日時点で，NVIDIA GeForce RTX 4060 Ti を 1 基用いて，コンピュータ将棋連続対局場所 floodgate[3] で R4200 弱

Player Statistics: nshogi_mini (floodgate)

Current Status (2 weeks)

rating (Δday, Δweek)	approximated rating in shogi club 24	wins (Δday)	losses (Δday)	%	last played
4194 (+4, -33)	3888	160 (+0)	96 (+0)	0.625	2025-03-12 13:44:26.000000000

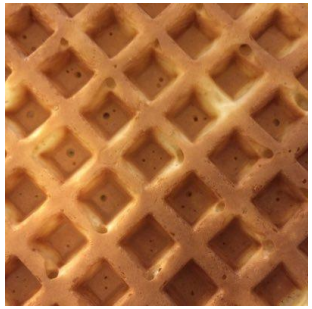
[3] http://wdoor.c.u-tokyo.ac.jp/shogi/floodgate.html

EOF

HoneyWaffle

第35回
世界コンピュータ将棋選手権
アピール文書
開発者　渡辺 光彦

開発者



氏名: 渡辺 光彦

職業: プログラマー

棋力: 将棋ウォーズで2級、ぴよ将棋でR900-1000程度の振り飛車党

Twitter: @shiroi_gohanP (https://twitter.com/shiroi_gohanP)

ニコ生の電王戦をきっかけにコンピュータ将棋を始める。

将棋連盟Liveやニコニコ生放送、AbemaTVの将棋中継が好き。

note書いています！→ https://note.com/honeywaffleshogi

文春オンラインのインタビュー記事→ https://bunshun.jp/articles/-/14921

🆕朝日新聞デジタルのインタビュー記事→ https://www.asahi.com/articles/AST19165DT19UCVL04KM.html

🆕教材にのりました！※→ https://x.com/shiroi_gohanP/status/1902639723673481229

※教科書ではありません。2日目に実物を持っていく予定です。

HoneyWaffle (ハニーワッフル) 名前の由来（2023年版）

・四角いワッフルは将棋盤と似ている

・ゆるふわスイーツ的なスナック感覚の軽さを表現

　元々タブレット向けに開発していたので物理的に軽いこと、振り飛車の軽い捌きができるようになるといいなという想いから命名しました。

　コンピュータ将棋といえば、人名 +将棋と命名するのが格調高いと思っています（森田将棋とか）。私が有名ではなく、将棋界で渡辺といえば渡辺明先生なので、「渡辺将棋」とは命名すべきではないと思いました。ということで、渡辺がだめなら光彦 →みつ→Honey、Waffleは上記のとおり将棋盤の意味。 いい命名じゃないです か？ すごくないですか？

以下のリンク先で出せるものは公開しています。使い方がおかしいのはいつものこと。

https://github.com/32hiko

コンセプト

青字は使用予定ライブラリ

「振り飛車で戦い続ける」

（1）振り飛車定跡

2024年度版からはあまり大きな変更はありません。先後ともに、複数の戦法を用意しておき狙いを絞らせません。

（2）評価関数：公開されている最強クラスの評価関数に対向形の棋譜から追加学習させたもの（予定）

現時点だと水匠10beta2の予定。※振り飛車評価関数として公開された「振電3」は私が使うわけにはいきません

（3）エンジン：やねうら王の最新版を魔改造したもの（予定）

振り飛車の評価値が理不尽に低い点の修正に改めて挑戦します。

（4）マシンはシングルインスタンスの予定

最後に

■コンピュータ将棋の振り飛車界隈では明るいニュースがありました。水匠のたややん氏が「振電3」を公開した件です。公開されたのは、おそらく振り飛車居飛車関係なくもっと強いものを出せる見通しだからかな、と推測しました。それはともかく、強い振り飛車評価関数が公開されたのはユーザーにとってありがたいことだと思います。選手権が終わったら、私のGitHubも振電3へ誘導するような感じに更新しておこうと思います。

■最近、プロの将棋界では振り飛車が見直される流れになっているかと思います。ただ、その理由としては相居飛車の後手番が大変だというのが大きいかと思いますが。コンピューター将棋界の飛車を振らない勢には、後手番で何かおもしろい工夫とかないのかな？というのを毎回期待していますが、難しそうですかね？

■コンピュータ将棋界でも、振り飛車対策というのをごくまれに見聞きすることがあります。少数派である振り飛車の対策をするくらいなら、上述の後手番対策をやる方が全然いいと思いますよ。断っておきますが、HoneyWaffle相手に変な手順で飛車を振らせないようにするのは、振り飛車の棋譜が減ることになるので本気で軽蔑します。相振り飛車のような駆け引きなら歓迎します！

■メンタルをやられてから毎回書いていますが。楽しくやれるうちはコンピュータ将棋と振り飛車を続けていくつもりですが、仮に今大会が最後になってしまっても悔いのないようにがんばります。

Ari Shogi and フレンズ

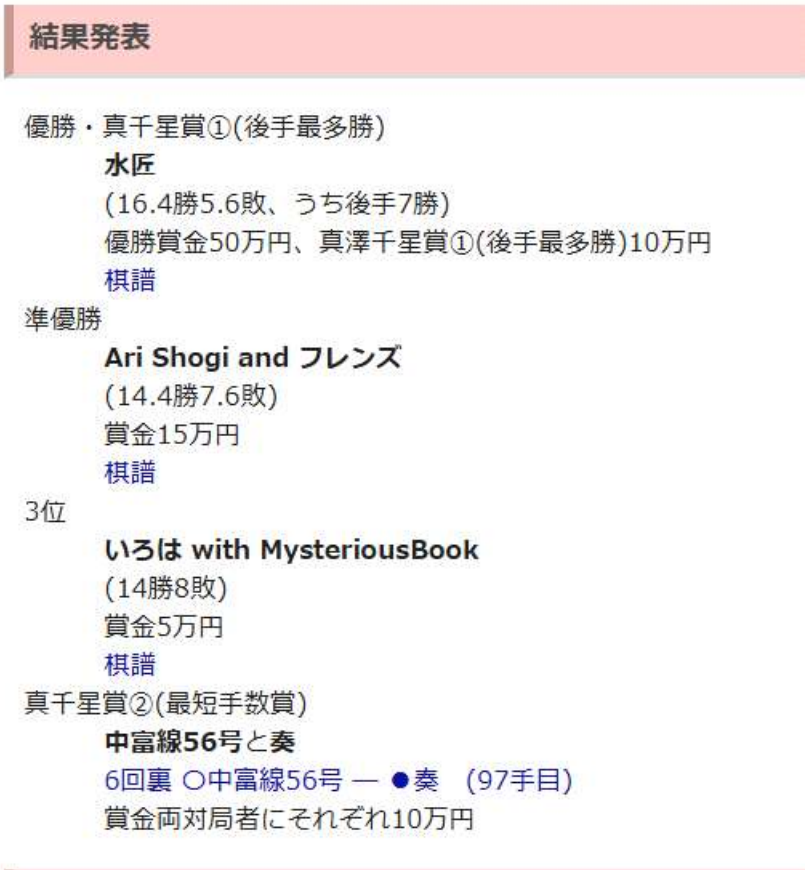
WCSC35 アピール文書(仮)

兵頭優空

概要

Ari Shogi and フレンズは、DL系AIとNNUE系AIを組み合わせる事で、低コストで多くの状況に対応できる事を目指しているハイブリッド型の将棋AIである。

2025年2月に開催された第3回世界将棋AI電竜戦ハードウェア統一戦(以降HWT3 | https://denryu-sen.jp/denryusen/dr5_hardware3/dr1_live.php)では、敗勢からの異常勝ちなどの幸運もあり、準優勝する事ができた。(異常勝ちの1局がない場合は3位)



結果発表

優勝・真千星賞①(後手最多勝)
水匠
(16.4勝5.6敗、うち後手7勝)
優勝賞金50万円、真澤千星賞①(後手最多勝)10万円
棋譜
準優勝
Ari Shogi and フレンズ
(14.4勝7.6敗)
賞金15万円
棋譜
3位
いろは with MysteriousBook
(14勝8敗)
賞金5万円
棋譜
真千星賞②(最短手数賞)
中富線56号と奏
6回裏 ○中富線56号 — ●奏 (97手目)
賞金両対局者にそれぞれ10万円

https://denryu-sen.jp/denryusen/dr5_hardware3/dr1_live.php

本大会に向けては、上手くいったHWT3の路線を継承しつつ、いくつか新しい事を試そうと考えている。

また、HWT3では、定跡を一切搭載しなくても割と戦えたので、今回も定跡なしで参加しようと考えている。

アピール文書は後日書き換えて差し替える予定

アピール文書の要約

「上手くいったHWT3の構成をベースに、DL系評価関数を重点的に改良する予定。具体的には、SAMを導入したり、ネットワークアーキテクチャの改造したり、ロマン溢れるカスタムoptimizerを作ったりしている。」

目次

第5回電竜戦本戦の構成など

HWT3で使ったものは第5回電竜戦本戦(以降DR5 | https://denryu-sen.jp/denryusen/dr5_production/dr1_live.php)で使ったものと大体同じなので、まずはそれの構成。

全体は

1. DL系AIとNNUE系AIに局面を送って思考させる

2. DL系AIが指し手を返したらNNUE系AIの思考を止める。

3. 必勝の手を見つけた場合はその手を返す。DL系AIの最善手とNNUE系AIの最善手が一致する場合もその手を返す。

4. NNUE系AIの評価値が規定値を超えているならNNUE系AIの手を採用する。(一部条件を満たしている場合除く)

5. NNUE系AIに「現在の局面にDL系AIの最善手を適用した局面」を送って短時間(1秒以下。条件によって変動)思考させ、評価値を取得する。

6. 「NNUE系AIの最善手を指した時の評価値」 - 「5で取得した評価値 * -1」が規定値(入玉模様の時や対局開始から既定の手数以内の時は少し大きめにする)を超えた場合はNNUE系AIの指し手を、それ以外の場合はDL系AIの指し手を返す。

という手順で思考。

DL系AIはpython-dlshogi2ベースの探索部に、ResNet15ブロックの評価関数(若干構造が異なるがdlshogi_dr2_exhiに近い)2つをアンサンブルした評価関数を搭載したものを採用。

NNUE系AIは、合議における役割が「DL系AIの補佐」や「優位な状況から勝ち切る」であり、「単体で最強」ではないので、単体性能を多少犠牲にしてもあらゆる局面で無難に強いものを目指した。

アーキテクチャは枯れた標準NNUE、ゼロから学習させるのは資源の都合もあって無理だったので公開されているものからの追加学習で用意。

最終的には、Háoから追加学習で作られたHaojianシリーズ(名前の由来はベースになっているHáoと、某ゲームに出てくる災いの剣)と、tanuki-wcsc28、BLOSSOM、水匠5、およびそれらから派生した評価関数を、連続対局の成績が上がるようにニューロンごとの混ぜ合わせ比率を進化的アルゴリズムで探したものを採用した。

定跡は

・定跡だけで決着がつくのは好きではない

・半端な出来の定跡や、評価関数との相性が悪い定跡は無い方が強くなりそう

・大会までに強い定跡を用意できそうにない

といった理由から採用しなかった。

第3回電竜戦ハードウェア統一戦の構成など

DR5のものをベースに、

・DL系評価関数に追加学習

・DL系探索部をpython-dlshogi2ベースのものからふかうら王に変更

・NNUE系のFV_SCALEを調整

・(DL系探索部の変更に伴い)MultiPVを使って意図的に手を散らす機能を追加

・(DL系探索部の変更に伴い)確率的PonderでNNUE系のPonder手を調べる優先度を上げる機能の廃止

などの変更を行った。

定跡に関しては、「定跡を作る資源で評価関数を強化して、本番のマシン・持ち時間で考えたほうが強くなりそう」と思ったので搭載していない。

色々やったのだが、提出前のfloodgateでの計測ではそこまで強くなっていなかった。

大会後のfloodgateでの計測でも、過去バージョンに比べて劇的に強くなっている感じではなかった。

大会前は、上述のようにあまり強くならなかった事と、

・GitHub版のやねうら王は支援者のみが入手できる開発版のやねうら王に比べ棋力が劣る

・大会では開発版やねうら王を利用するチームが多い

・ここ最近、NNUE系の強さが全体的に跳ね上がっているが、それらに完全に置いて行かれている

・定跡が強いと言われるルールだが定跡を一切搭載していない

といった理由から苦戦すると思っていたが、善戦して上位に行けたので、自分でも

驚いている。

理由は色々あると思うが

・持ち時間 / ハードウェア構成と、評価関数の重さがかみ合っていた

・「DL系で優位を作りNNUE系で押し切る」という作戦が割と上手くいった

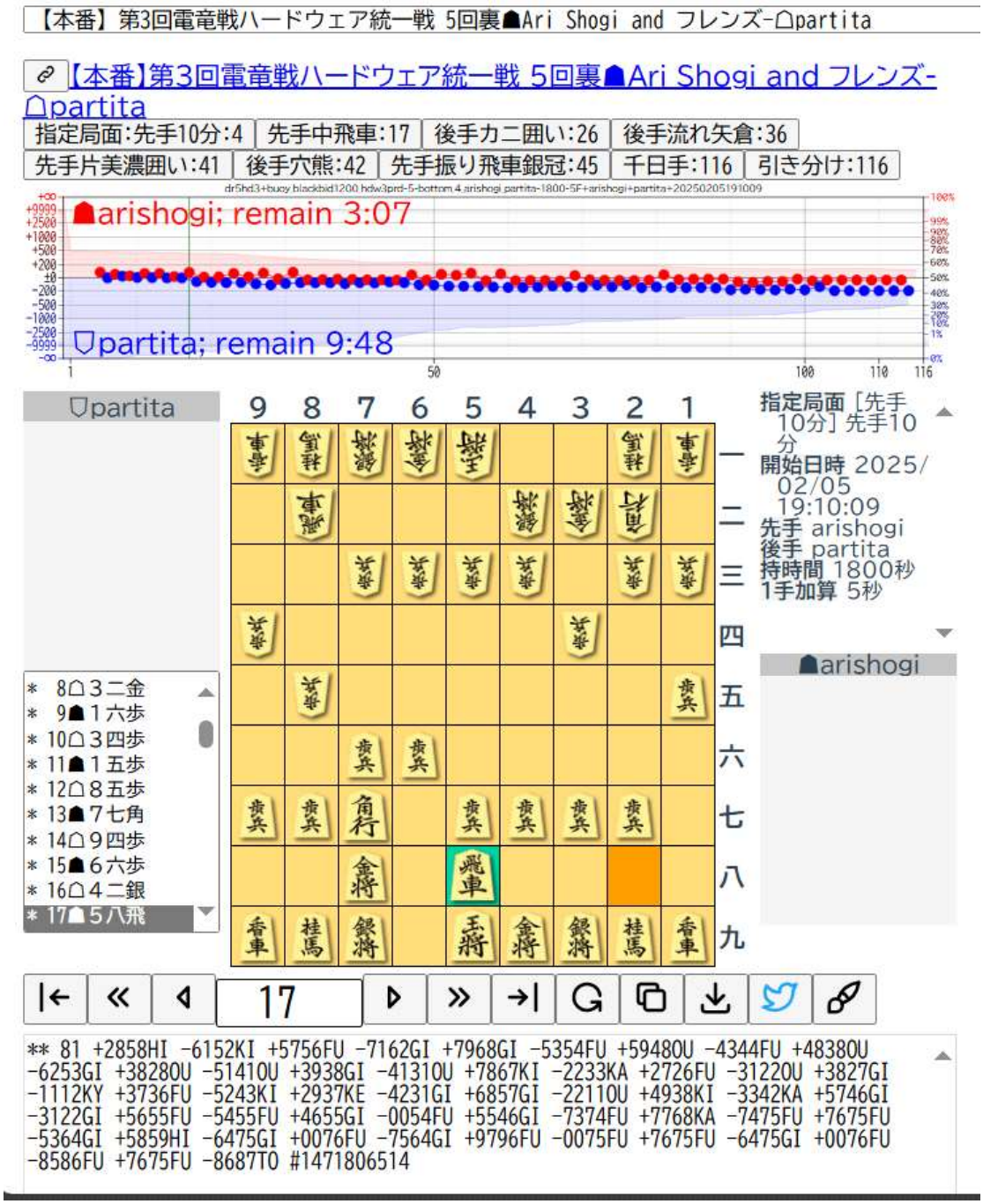
・定跡を使わないことでトラップのような変化を避けれた

・単純に運がよかった

あたりが上位に行けた原因なのではないかと考えている。

今回の結果で、合議部分などは悪くなさそうだという事が分かったので、WCSC35に向けては評価関数などの強化を重点的に行おうと考えている。

一応、序盤に意図的に手を散らす機能のランダム性が強すぎて、HWT3で「普段は指さない振り飛車を指して先手千日手になる」という現象が発生したので、そこは改善したいと考えている。



https://denryu-sen.jp/denryusen/dr5_hardware3/dist/#/dr5hd3+buoy_blackbid1200_hdw3prd-5-bottom_4_arishogi_partita-1800-5F+arishogi+partita+20250205191009/17

別に勝てるなら戦法は何でも良いんだが、これ以外の対局で振り飛車を指したのは見たことがないので、指しこなせるイメージはない。実際、大事な先手番で千日手になってしまっている。

恐らく

・DL系評価関数の学習にAoba振り飛車の棋譜も含まれている

・NNUE系評価関数の材料のいくつかは実験で作った振り飛車評価関数

からそこまで振り飛車を低く評価しない事と、ランダム性で飛車先の歩をつく手が選ばれなかった事と、飛車を振るのが良いと判断する形になっていた事で発生したのではないかと考えている。

取り組む事

・SAM(Sharpness-Aware Minimization)の導入

昨年、ponkotsuチーム(https://www.apply.computer-shogi.org/wcsc34/appeal/ponkotsu/ponkotsu_appeal_2024.pdf)も採用していた手法。

1. 今のパラメータAの周辺で最も損失が悪化するパラメータBを求め

2. パラメータBで損失を求め

3. 2の損失に対する勾配と、適当なoptimizer(MomentumSGDとか)を使って、パラメータAから更新する

という流れでパラメータを更新する事で、損失が小さく、さらにその周辺が平坦なパラメータを求めるという手法。

こんな面倒な事してまで周辺が平坦なパラメータを探すのは、そういったパラメータは汎化性能が高い事が多かったり、量子化などとの相性が良いなど、いろいろ良い点がある事が知られているから。

将棋AI的にもそういったパラメータは嬉しいので、コストをかけてでも採用する価値があるのではないかと考えている。

周辺が平坦なパラメータ云々については、「深層ニューラルネットワークの高速化」という本に色々書いてあったので気になる人は読んでみると良いと思う。

上の本と同じ人が公開している

https://speakerdeck.com/joisino/landscape

にも詳しく書いてあるので興味のある人は見てみると面白いと思う。

・optimizerの改造

まだ途中なので、そこまで書く事がないが、既存のoptimizerの集団（ Adam、Santa、Lionなどなど)から進化アルゴリズムを使って将棋AIの学習に適したoptimizerを作るという実験をやっている。

1. 非常に簡単なタスクでエラー無しで一定のスコアが出るものを選別

2. 実際の利用に近い条件で軽く学習させてみてスコアを確認

3. スコア上位のoptimizerを優先的に使い、次世代のoptimizerを作る(優秀なoptimizerはそのまま引き継ぐ ）

という流れ。

3/30現在、一通りサイクルが回せるようになったので細部の調整や初期集団のoptimizerの追加などを行っている。

正直、このリソースでSGDとかAdamみたいな実績のあるoptimizerを使って大きいモデルを学習させたり、教師データを生成して学習させた方がずっと強くなるだろう。

しかし「将棋AIの学習に適するように進化したoptimizer」はロマンに溢れている(個人の感想です)。

趣味でやっている事なのでロマンはしっかり追求したい。

ので、4月の前半くらいまではこれに取り組もうと考えている。

・新しいDL系評価関数の学習

上記の改造の他、

・アンサンブルモデルを使った蒸留

・dlshogiなども導入しているTransformerを取り入れたネットワーク構造

・左右反転などのデータ拡張

などを使って新しい評価関数を学習させる予定。(すべて実装済みで動作テスト済み ）

ネットワークサイズはdlshogiの公開モデルと同等レベルの15ブロックか

ら、より高い精度を求め20ブロック以上にする予定。

・教師生成

今までは公開教師データのみ使っていたが、今回は遊んでいるCPUを使って少し教師データを生成したので、それも使う予定。

floodgateの上位の対局やWCSC、電竜戦の対局から抽出した序盤の局面からファインチューニング用の高品質データを、昨年技巧チーム（ https://www.apply.computer-shogi.org/wcsc34/appeal/Gikou/gikou_appeal_detail.pdf ）が使っていた「将棋81万」を参考にした開始局面から、標準NNUEのdepth9で事前学習用の教師データを生成している。

・NNUEの強化

現状では標準NNUEを使う予定。

しかし手元の標準NNUE評価関数は、学習しても精度がなかなか伸びない上に、「テストデータに対する精度は少し上がったが連続対局で強くなっていない」という現象も発生しているため、ほとんど強くならないと思う。

一応、伸びしろのない評価関数のパラメータを弄って無理やり伸びしろを作るアイディアがいくつかあるので、余裕があればそれを試す予定。

・探索部の改造

余裕があったらやねうら王、ふかうら王の改造を行う。(あまり余裕がないのでやらない可能性が高い)

使用ライブラリ等

・やねうら王、ふかうら王

https://github.com/yaneurao/YaneuraOu

用途：DL系、NNUE系の探索部

採用理由：オープンソースでかつ強いため

・python-dlshogi2、cshogi

https://github.com/TadaoYamaoka/python-dlshogi2

https://github.com/TadaoYamaoka/cshogi

用途：合議エンジンのベースや学習部など多くの場面

採用理由：便利で扱いやすいため。これらを利用したコードが手元に大量にあるのでそれを活かすため。

・dlshogi

https://github.com/TadaoYamaoka/DeepLearningShogi

用途：学習部などを参考にしている他、一部のスクリプト(教師データの変換、探索部パラメータの自動チューニングなど)を利用

採用理由：学習部は多くの人が利用しており実績があるため。スクリプトは便利なものが多く公開されており自分で作る手間が省けるため。

・NNUE-Pytorch

https://github.com/nodchip/nnue-pytorch

用途：自前のNNUE学習部で色々参考にしている

採用理由：実績があるから

ちなみに自前のNNUE学習部は公開している。

https://github.com/YuaHyodo/Ari_Shogi_NNUE_train

ずっと放置していて手元のバージョンとの差分が結構あるので気が向いたときにちょっとずつ反映させる予定

・shogihome

https://github.com/sunfish-shogi/shogihome

用途：CSAサーバーへの接続など

採用理由：動作が安定していて、実績もあり、自分の作ったスクリプトと異なり見た目も良いので。

利用した教師データ

教師データは生成が大変なので基本的に公開されているものに頼っている。

公開されており、多くの人が利用するデータでも、

https://huggingface.co/datasets/nodchip/shogi_hao_depth9

などはSSDの容量が足りないので現状使用していない。

・AobaZero、Aoba駒落ち、Aoba振り飛車の教師データ

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

http://www.yss-aya.com/komaochi/index.html

http://www.yss-aya.com/furibisha/

・水匠開発者のたややん氏の公開している教師データ

https://drive.google.com/drive/folders/1IVp_xcNW-XfqypirLrtVCEKN2OvAfe2t

・dlshogi with GCTの教師データ

https://tadaoyamaoka.hatenablog.com/entry/2021/05/06/223701

・書籍「強い将棋ソフトの創りかた」の教師データ

・Qhapaq Pretty Derbyの教師データ

https://qhapaq.hatenablog.com/entry/2021/11/23/220251

・floodgateの棋譜

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

NNUE評価関数のベースとして利用した公開モデル

・Háo(https://github.com/nodchip/tanuki-/releases/tag/tanuki-.halfkp_256x2-32-32.2023-05-08)

・BLOSSOM(https://x.com/senninha_a/status/1654675205464390)

・水匠5(https://github.com/mizar/YaneuraOu/releases/tag/v7.5.0)

・tanuki-wcsc28(https://github.com/nodchip/tanuki-/releases/tag/wcsc28)

採用理由: 無料で利用できかつ強いため。実験の過程でできた派生評価関数が手元に大量にあるため

おまけ

1. 終盤に評価値がガタガタする理由

Ari Shogi and フレンズでは、合議の結果を出力するとき、「DL系AIとNNUE系AIの最善手が一致する場合」と「DL系AIの最善手が採用された場合」はDL系の評価値が、それ以外の時はNNUE系の評価値が出力されるようになっている。(ちなみにDL系の評価値は偶数になるように、NNUE系の評価値は奇数になるように細工している)

終盤にグラフがガタガタするのは、終盤は基本的にNNUE系の評価値が出力されるが、終盤でも一部の状況(指し手の一致や、入玉などの条件を満たしている時)だとDL系の指し手が採用されて、NNUE系のそれとはスケールが異なるDL系の評価値が出力されるから。

2.評価関数の精度計測結果

GCTのノートブックのテストデータ(参考: https://tadaoyamaoka.hatenablog.com/entry/2021/09/11/155938)に対する精度

方策正解率 / 勝敗正解率 / 価値損失 / 方策エントロピー

DR5(2日目)で使ったやつ

0.5196 / 0.7672 / 0.4625 / 1.1599

HWT3で使ったやつ

0.5211 / 0.7649 / 0.4619 / 1.3228

HWT3で使ったやつ単体での精度

1つめ: 0.5146 / 0.7600 / 0.4679 / 1.3450

2つめ: 0.5125 / 0.7602 / 0.4702 / 0.9996

アンサンブルしてもdlshogi dr2 exhiに比べると若干精度が低い。

3/31時点の最新モデルでアンサンブルしてようやくdlshogi dr2 exhiを上回れる、といった精度。

4月半ばくらいから本格的な精度向上を行い、選手権本番ではdlshogi dr2 exhiを大きく上回れるようにしたいと思っている。

アピール文

2025.2.23作成
花井祐

プログラムの名称：いちびん
プログラムの特徴：

1. 概要

2023.2024年のバージョンとほとんど同じです。NNUEを基本、自作の教師データで学習、やねうら王のラーニング機能を使って思考部分を作成、探索は、やねうら王の探索ルーチンを使用

1-1. 教師データ

三駒関係に基づく評価関数（Daphin）をもとに、2023年当時のやねうら王を使って教師データを生成、その際に、ソースコードに手を加えて、1手あたり最大で100万局面、探索の枝狩りをゆるくしています。

1-2. 評価関数

やねうら王の探索を考慮して（上述）約15億局面ほどを作成、それを用いて、全くのヴァージンから、評価関数を作成、パラメータを思考錯誤して、数十の試作を完成させ、その中からこれぞというものを選びました。

1-3. 定跡

2023年の参カロバージョンを大本にして、約二年間、ひたすら続けて創作しました。

以上。

第35回世界コンピュータ将棋選手権 アピール文書
プログラム名「タンゴ」
開発者　渡邊敬介
2025年1月19日

概要

実現確率探索による深い読みと、機械学習により最適化された正確な評価関数により、強力なコンピュータプレイヤの実現を目指します。

本プログラムの大きな特徴は、局面評価関数および実現確率用の着手確率にFactorization Machinesを使用している点です。2023年以前のコンピュータ将棋選手権でこのような手法を用いていたチームは、私の知る限りでは私のチーム以外に存在しません。

局面評価関数

本プログラムでは、駒の損得以外に局面評価関数の特徴量に下記の3つを採用しています。先後の対称性を保つため、先手から見た盤面と後手から見た盤面それぞれについて下記の特徴量を入力とするFactorization Machineによりスコアを計算し、算出された先後のスコアの差と駒の損得を合算してその局面の評価値としています。

1. 2駒の位置関係 (俗に言うKP + PP + KK)
2. 自玉周辺25マスの双方の効き
3. 手番

Factorization Machinesを使用することで、これらの入力特徴の組み合わせ特徴を取り込んだ極めて表現力の高い評価関数となっていると考えています。例えば、2駒の位置関係同士の組み合わせは4駒の位置関係(俗に言うKKPP + KPPP + PPPP)に相当します。

実現確率探索用の着手予測

Factorization Bradley-Terry モデル[1]を使用しています。通常のモデルと小規模で高速な簡易計算用モデルの2種類を用意し、探索中の末端付近ではこの簡易計算モデルを使用することで高速化を図っています。

また、Factorization Bradley-Terry モデルでは着手確率を計算するためにはその局面の全ての着手のスコアを計算した上でソフトマックス関数に通す必要があります。そこで、「駒の位置」や「王手がかかっている」などの盤面共通の特徴量を最初に計算し、各着手のスコア計算で再利用できるようにしています。さらに、駒の位置については差分計算可能なので、簡易計算モデルで使用される駒の位置の特徴量については差分計算を行っています。

追試可否

　再現実験を可能とするため、対局用プログラムだけでなく学習に使用したデータセットも大会後1年間保管する予定です。

参考文献

[1]Xiao, Chenjun, and Martin Müller. "Factorization ranking model for move prediction in the game of Go." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. No. 1. 2016.

「習甦」

【α-β探索とMCTSにおけるデータ共有による性能改善】

探索では、MCTSをメインエンジンとし、α-β探索のハッシュテーブルを以下のように活用

・Policyの調整：最善手はPUCTのパラメータを大きくして、訪問回数を増加させる

・Valueの調整：格納された値が閾値を超えている場合、確率的にリーフノードとする

機械学習では、以下の公開棋譜データを使用

・AobaZero

・dlshogi

・水匠

・floodgate

中富線 56 号アピール文書

開発代表：横井悟郎

0.はじめまして

初出場である。まずはエラー無く完走したい。

1.開発動機

作者の専攻は合成生物学である。作者の夢は，生体分子で作ったニューラルネットワークを生物に組み込み，将棋の手を考えさせることである。他方，昨年度は AlphaFold がノーベル化学賞を受賞したことで界隈でも大きな話題を呼んだ。AlphaFold を開発した Google DeepMind は AlphaGo，AlphaZero などのボードゲーム AI で知られるディープラーニング企業であり，「ボードゲーム AI の開発ノウハウを生物学へ応用」させた 1 つの成功例であると言える。作者もそれを目指し，ニューラルネットワーク/DNN モデルへの理解を深めることを目的に開発を始めた。

2.使用ライブラリ

yaneurao/YaneuraOu：探索部に使用

nodchip/nnue-Pytorch：評価関数作成に使用

kobanium/aobazero：知識蒸留に使用

3.学習手法

AobaZero を用いて作成した教師から nnue-Pytorch を用いて GPU で学習させている。

技術的な工夫としては，出来上がったモデルの精度評価方法の独自化，自動・手動を組み合わせた定跡生成，使用ハードウェアに最適化したネットワークアーキテクチャの選定などが挙げられる。

4.設計思想

コンセプトとして「自分の得意なパターンに持っていく」ことができるモデルを目指している。自分の得意パターンに引き込むことは，人間・AI 問わず，将棋において勝率を上げる有効な方法であると作者は信じている。機械学習の分野では「あるタイプの問題に特化したモデルは，汎化性能の高いモデルに比べて，その問題領域では高い性能を発揮できる」という，ノーフリーランチ定理と呼ばれる性質が知られているが，限られた教師量/計算資源/表現力の中である問題に対する性能を最大化するには，問題の性質に最適化する，あるいは，問題そのものをモデルの性質に寄せるというような戦略が有効であると考えられる。殊に NNUE モデルは，元々が推論速度を重視して軽量な構造になっているために，モデルが持つ表現力のポテンシャルが限られており，将棋のゲーム木全体をカバーする汎化能力を持たせることが難しいと思われる。そのため，実際には，教師生成の段階で，実現確率が低い戦型の教師局面を削除するといったことが行われる。NNUE 系のモデル同士の対局が，特定の戦型（角換わり）に大きく偏るのは，このような，少々biased な学習を行っているためであるとも考えられる（出来上がったモデルの性能評価，取捨選択も自己対局で行うことが多

い)。そのため，NNUE モデル同士の対局で勝率を上げるためには，特定の問題領域（戦型など）に強いモデルを作り，その問題領域に相手を引きずり込むのが有効になると考える。拙作では，この設計思想に基づき，学習されたモデルの性能評価。取捨選択を特定の戦型の局面からの自己対局結果を利用し，さらに定跡部分でその戦型に誘導する，という作戦を採用している。

3.定跡作成手法

特化させる戦型は，概ね作者の棋風に合わせている（これによって作者にとって最良の教師を作成し作者自身を強化学習させようという密かな野望を持っているのは内緒である)。ある程度作戦が定まってからの部分は自動生成を行っている。最序盤の戦型選択や，合流手順の検討などは，人間の方が上手だったりするのは興味深い。

5.謝辞

将棋 AI に関する様々なライブラリや資料を公開してくださっている開発者の皆さん，WCSC に参加する動機を与えてくださった，独創的なアピール文書を書いている開発者の皆さん(きふわらべさんなど)，プログラミングに興味を持たせていただいた研究室の先輩方と先生方に，厚く感謝申し上げます。

やねうら王 PR 文書

■ 定跡について

やねうら王では、今回『新ペタショック定跡』を搭載している。

これは、1 局面につき 2 億ノード以上探索して生成した定跡である。探索したあと定跡をペタショック化している。2025 年 3 月 23 日時点で 80 万局面程度ある。大会当日までには 150 万局面程度になると見込まれる。

ペタショック化とは、定跡ツリーを minimax 化したものが局面数 N の定数倍の時間で得られる非常に優れたアルゴリズムである。このアルゴリズムの詳細についてはどこかに論文を書く。この実装は、やねうら王の GitHub のコードにある。

やねうら王の GitHub
https://github.com/yaneurao/YaneuraOu

■ 探索について

あとで書く。

■ 評価関数について

あとで書く。

Kanade アピール文書

山口　奏
2025 年 3 月 31 日

Kanade は一般的に DL 系と呼ばれる類の将棋 AI です。
主に評価関数と定跡を独自で作成しており、探索部はふかうら王を使用させていただいております。

評価関数

Resnet25x320 の標準的なネットワークを使用しています。
序盤中盤に比べ、終盤の評価精度が著しく悪い傾向があります。
教師データは、公開されているデータのみを使用しており、独自で生成した教師データは使用していません。どの教師データが学習に有効的かを研究し、厳選したデータを使用しました。教師データを水増しするために、局面を左右反転させたものも教師データに使用しています。
また、正規化やバッチノーマライゼーションに関する技術などの学習テクニックを多数実験し、最も実験結果が良かった数値やテクニックを採用しています。

定跡生成

4 つの NNUE 系のソフトを使用して対局を行い、対局結果を min-max で親ノードに伝播させる手法で定跡を生成しています。
4 種類のソフトを使用することによって、定跡の穴をより発見しやすくしています。
定期的に人間が定跡の精度を確認し、間違った評価をしていると思われる局面が見つかった場合は、その局面を対局開始局面に設定することで正しい結論に導きやすくしています。

WCSC35 W@nderER アピール文書

Dated: 2025/03/31
Hiromitsu SAKURAI

これまでと同様に、入玉宣言による勝利を積極的に目指します。

前回大会からの変更点

- 昨年のWCSCと電竜戦にて行ったNetwork構造の試行錯誤を行っており、7x7のラージカーネルからRyfcNetのC-layer風のAsymmetry factorized畳み込み（疑似ラージカーネル）として1x9と9x1のカーネルに変更
- 入玉特徴量を導入
- Floodgateにおける入玉が発生した棋譜をベースにして教師を生成

Phonanza
アピール文書 WCSC35

開発者 ペンギンクミマヌ

2025年3月28日　作成

プログラム情報

プログラム名:Phonanza

プログラム名の由来:　歴史に名を刻んだあのソフトをリスペクトして名付けた。あのソフトのようにトップに君臨し、将来的に将棋AIの大会で四冠を目指すつもりである。

使用ライブラリ: nnue-pytorch、やねうら王、cshogi,dlshogi

採用している手法: DL系モデルによるNNUEの学習データの評価値付け替え

探索部:やねうら王

コメント

WCSC34以後より、NNUE系評価関数の作成を開始しました。nnue-pytorchによる学習、評価値付け替えといったnodchipさんの工夫を全面的に取り入れ開発しています。アピール文書作成時点において、振電3(振り飛車党)に勝率9割以上を叩き出す強い評価関数ができたはずだった...floodgateに参加したところ、レートが3700しかなく振電3キラーでしかなかったことが判明し、WCSC35までにどのような改良を施すか苦慮しているところです。

またWCSC34の時は定跡を搭載しても、それを生かせる評価関数がなかったので、搭載を見送っていたのだが、今大会は定跡を搭載予定です。

参考文献

・nnue-pytorchの環境構築（将棋AI評価関数学習器） - select766’s diary

・やねうら王Wiki

開発者情報

開発者名:ペンギンクミマヌ　(本名)山下公誠

SNS: YouTube X note

2024年の電竜戦に引き続き、深層学習モデルの効率的なアンサンブルと棋風調整の研究を行います。

2024年の電竜戦では、メモリ6GBのゲームノートPCで対局＋振り飛車縛りという厳しい条件にも関わらずA級リーグ入りを達成したことを鑑みると、深層学習のアンサンブルには将来性があると考えています。

今年はノートPC参加に賞金があると聞いたので全力でノートPC最強の座を取りに行きます！

将棋などのゲームAIにおいて、アンサンブルの有効性は至るところで示されている一方で、限られた計算資源で効果的なアンサンブルを行うにはいくつかの技術的な課題を解決しなければいけません。将棋AIであれば

・時間制御をどうするか
・ponderをどうするか
・MultiPVなどの複数手表示を行えるような設計は探索効率の低下をもたらす（やねうら王などのmin-max木を使う場合は特に）
・そもそも計算資源がもったいない（MultiPonderは極めて有効な手段である一方で並列計算はコストが高い）
・アンサンブルの比率は必ずしも1:1にしたいわけではない（重み0.3ぐらいがいいみたいな場合をどうするか）

といった問題があります。前回の電竜戦では複数の深層学習モデルの合議方法を見直すことで単一のGPUで動くアンサンブル手法を開発しました（現在論文執筆中）。

今回のWCSCではこれに加えて

・特定の棋風を狙って出すことができないか

に挑戦します。振り飛車をやらせる代表的な方法は定跡などで制限をかけることですが、探索のルールベースで特定の棋風を無理やり指させる手法も試みられています

例：最強の振り飛車ソフトを作りたい！
https://saihyou.hatenadiary.jp/entry/2025/02/27/200039

しかし、こうしたアプローチは人間が普段考えているようなより柔軟な棋風調整にはマッチしていません。

こうした問題を解決するには、特定の棋譜を起点にして、その棋風への類似度を評価値へのボーナスとするようなアプローチが有効であると考えられます。

しかし、評価値へのボーナスは探索部に埋め込むと探索速度の低下を招きます。また、探索終了後に手を採択する際の補正校として用いるとするならばMultiPVへの対応が必須となります。

こうした問題をバランス良く解決し、特定の棋風を効果的に指しこなすようなルーチンの開発を目指します（失敗したらゴメンナサイ）

第35回
世界コンピュータ将棋選手権
なのはアピール文書

2025年3月31日 川端一之

■**なのは**ってなんだよ

- 熱血魔法バトルアクションアニメ「魔法少女リリカル**なのは**」シリーズの主人公**高町なのは**を由来にし、さまざまな称号を冠する彼女のような強さを盤上で実現したいという願いを込めています。

- 「名前の割に強い」という声をいただきますが、その認識は逆で、「名前負けしている」や「名前の割に弱すぎる」というほうが妥当な評価です。

■作者はどんな人？

- 静岡県出身 愛知県在住
- とあるメーカーに勤務(ちょっとAWS使う)
- 好きな食べ物は焼肉、しゃぶしゃぶ、寿司
- 好きなアニメは魔法少女リリカルなのは、りゅうおうのおしごと！、痛いのは嫌なので防御力に極振りしたいと思います。、くまクマ熊ベアー、とある科学の超電磁砲、ラブライブ！、五等分の花嫁

  最近は「お隣の天使様にいつの間にか駄目人間にされていた件」がお気に
- 将棋ウォーズ 1級
- 囲碁は日本棋院 初段(アマチュア)

■開発環境

- こんなPCで開発しています

  Lenovo Thinkpad T14
  CPU：AMD Ryzen 7 PRO 4750U
  プレゼントでいただきました

- 出場PC

  Minisforum UM790Pro
  CPU：AMD Ryzen 9 7940HS
  RAM：32GB
  OS：Windows 11 Pro

■なのはの構成

- MSYS2のg++で開発
- 手生成では歩、角、飛の不成も生成
- 探索はStockfish使用
- Bitboard未使用(盤情報は配列)
- 定跡部は実戦での出現数および勝率を考慮して手を選択
- 評価ベクトルは3駒関係(KPPT)

…と、数年前に見たような平凡な構成

■今回の変更点

- [予定]探索部の強化(Stockfish 8改→ Stockfish 15.1改)
- [予定]評価関数の強化(配布教師データでの学習)

■意気込み

- 現地会場から参加！
- 0次予選突破！
- 出来れば勝ち越したい！
- 詰めルーチンを間に合わせたい！

■今後の野望(?)

- 評価関数のNNUE化
- ミクロコスモスを解く
- 非コピーレフト系ライセンスを採用して開発

■使用ライブラリについて

- なのはmini　Stockfishをベースに独自に将棋化しました。

■使用データについて

- 評価ベクトルの学習には、一般に流布された教師データを使う予定
  - やねうらおさん配布の110億教師データ
  - nodchipさん配布の教師データ

■最後に

- **なのは**アピール文書は以上です
- 最後まで読んで頂きありがとうございます

絵： COCOさん

■参考文献

- 小谷善行、他:「コンピュータ将棋」, サイエンス社, 1990.
- 松原仁 編著:「コンピュータ将棋の進歩」, 共立出版, 1996.
- 松原仁 編著:「コンピュータ将棋の進歩2」, 共立出版, 1998.
- 松原仁 編著:「コンピュータ将棋の進歩3」, 共立出版, 2000.
- 松原仁 編著:「アマ四段を超えるコンピュータ将棋の進歩4」, 共立出版, 2003.
- 松原仁 編著:「アマトップクラスに迫るコンピュータ将棋の進歩5」, 共立出版, 2005.
- 池泰弘:「コンピュータ将棋のアルゴリズム」, 工学社, 2005.
- 金子知適, 田中哲朗, 山口和紀, 川合慧:「新規節点で固定深さの探索を併用するdf-pn アルゴリズム」, 第10回ゲーム・プログラミングワークショップ, pp.1-8, 2005.
- 脊尾昌宏:「詰将棋を解くアルゴリズムにおける優越関係の効率的な利用について」, 第5回ゲーム・プログラミングワークショップ, pp. 129-136, 1999.
- 保木邦仁:「局面評価の学習を目指した探索結果の最適制御」 http://www.geocities.jp/bonanza_shogi/gpw2006.pdf
- 岸本章宏:「IS 将棋の詰将棋解答プログラムについて」, http://www.is.titech.ac.jp/~kishi/pdf_file/csa.pdf, 2004.
- 橋本剛, 上田徹, 橋本隼一:「オセロ求解へ向けた取り組み」, http://www.lab2.kuis.kyoto-u.ac.jp/~itohiro/Games/Game080307.html

■参考Web

- やねうら王 公式サイト: http://yaneuraou.yaneu.com/
- 千里の道も一歩から: http://woodyring.blog.so-net.ne.jp/
- 小宮日記: http://d.hatena.ne.jp/mkomiya/
- State of the Digital Shogics [最先端計数将棋学]: http://ameblo.jp/professionalhearts/
- ながとダイアリー: http://d.hatena.ne.jp/mclh46/
- 毎日がEveryday: http://d.hatena.ne.jp/issei_y/
- Bonanzaソース完全解析ブログ: http://d.hatena.ne.jp/LS3600/
- aki.の日記: http://d.hatena.ne.jp/ak11/
- FPGA で将棋プログラムを作ってみるブログ: http://blog.livedoor.jp/yss_fpga/

※読めなくなったサイト含む

第35回世界コンピュータ将棋選手権

もふ将棋　アピール文書

東京大学原田研究室

日高雅俊、太田一毅、橋本智洋、原田達也

2025年5月2日

概要

- 新しい強化学習アルゴリズムにより、深層学習系評価関数をゼロから学習します
- iPhoneを計算資源として用いた分散計算により、プロ棋士に迫る強さの評価関数の実現を目指します

手法

- 評価関数の学習
  - ボードゲーム向けの新しい強化学習アルゴリズム*1により、人間の棋譜を用いることなくゼロから深層学習系評価関数を学習します
  - 学習アルゴリズムの変更により、新しい指し手や棋風が見られるかもしれません
  - 学習用の棋譜を生成する計算資源として、一般的に用いられる強力な計算機だけでなく、スマートフォン(iPhone)を用いた分散計算を実施しました
- 対局
  - ふかうら王*2ベースの探索エンジンをiPhone上で動作させます
  - 30台のiPhoneで10種類のモデルを動作させ、合議により棋力の向上を図ります

*1 学習アルゴリズムの詳細は、後日論文として発表予定
*2 https://github.com/yaneurao/YaneuraOu

合議の様子


5i6h
5i6h
3i3h
5i6h
2e2d
5i6h
5i5h
2e2d
3i3h
5i5h
5i6h
P*8g
5i6h
3i3h
5i6h
5i6h
3i3h
5i6h
3i3h
5i5h
5i6h
2e2d
5i6h
2e2d
2e2d
3i3h
2e2d
5i6h
5i6h
5i6h

合議の様子

もふ将棋

もふ将棋: △後手
自分の手番
残り時間: 先手16分10秒、後手13分49秒
現局面の消費時間: 171046秒

合議結果

△５二金	△８五歩	△９四歩
11 票	9 票	6 票

No. 1 思考中 評 42 **△４二玉** ▲３六飛△４五銀左 NPS 375/Nodes 3402	No. 2 思考中 評 33 **△４二玉** ▲３六飛△４五銀左 NPS 437/Nodes 3968	No. 3 思考中 評 77 **△５二金** ▲３六飛△４五銀左 NPS 414/Nodes 3765	No. 4 思考中 評 78 **△５二金** ▲５八金△８五歩 NPS 399/Nodes 3616	No. 5 思考中 評 38 **△９四歩** ▲５八金△８五歩 NPS 392/Nodes 3563
No. 6 待機中 評 22 **△８五歩** ▲３六飛△４五銀左 NPS 423/Nodes 3976	No. 7 思考中 評 28 **△８五歩** ▲９六歩△９四歩 NPS 429/Nodes 3891	No. 8 思考中 評 38 **△８五歩** ▲３六飛△４五銀左 NPS 432/Nodes 3916	No. 9 思考中 評 42 **△８五歩** ▲３六飛△４五銀左 NPS 423/Nodes 3845	No. 10 思考中 評 26 **△９四歩** ▲３六飛△４五銀左 NPS 418/Nodes 3800
No. 11 待機中 評 40 **△９四歩** ▲５八金△８五歩 NPS 422/Nodes 3976	No. 12 思考中 評 40 **△４二玉** ▲３六飛△４五銀左 NPS 409/Nodes 3720	No. 13 待機中 評 83 **△５二金** ▲５八金△８五歩 NPS 411/Nodes 3870	No. 14 思考中 評 39 **△８五歩** ▲３六飛△４五銀左 NPS 417/Nodes 3800	No. 15 思考中 評 33 **△５二金** ▲３六飛△４五銀左 NPS 399/Nodes 3623
No. 16 思考中 評 37 **△５二金** ▲３六飛△４三金右 NPS 425/Nodes 3862	No. 17 思考中 評 19 **△８五歩** ▲５八金△４二玉 NPS 400/Nodes 3632	No. 18 思考中 評 40 **△９四歩** ▲３六飛△４五銀左 NPS 420/Nodes 3816	No. 19 思考中 評 65 **△５二金** ▲３六飛△４五銀左 NPS 419/Nodes 3804	No. 20 思考中 評 42 **△４二玉** ▲３六飛△４五銀左 NPS 427/Nodes 3867
No. 21 思考中 評 65 **△５二金** ▲３六飛△４五銀左 NPS 430/Nodes 3905	No. 22 思考中 評 41 **△９四歩** ▲３六飛△４五銀左 NPS 374/Nodes 3395	No. 23 思考中 評 46 **△９四歩** ▲３六飛△４五銀左 NPS 398/Nodes 3744	No. 24 思考中 評 61 **△５二金** ▲３六飛△４五銀左 NPS 399/Nodes 3632	No. 25 思考中 評 52 **△８五歩** ▲３六飛△４五銀左 NPS 415/Nodes 3769
No. 26 待機中 評 20 **△８五歩** ▲３六飛△４五銀左 NPS 427/Nodes 4024	No. 27 思考中 評 74 **△５二金** ▲５八金△８五歩 NPS 419/Nodes 3806	No. 28 思考中 評 18 **△８五歩** ▲３六飛△４五銀左 NPS 416/Nodes 3784	No. 29 思考中 評 36 **△５二金** ▲３六飛△４五銀左 NPS 435/Nodes 3952	No. 30 思考中 評 40 **△５二金** ▲３六飛△４五銀左 NPS 425/Nodes 3855

「もふ将棋」の由来

- Mobile Federation (連合)
- 一つ一つの計算能力は強くない、モバイル端末（iPhone）が力を合わせて課題に取り組む様子を表現しています

芽生将棋のアピール文章

2025年

芽生将棋

去年将棋を始めて、将棋AIがやりたくてすぐにプログラミングも始めました。すると強い将棋AIを作る上で偉大な先人たちが公開している棋譜とNNUEのライブラリで学習を進めて、探索部と組み合わせるのが強くなると分かりましたが、実装してみるとこれはあまりにも高度すぎて、自分が改変してより強くするのは難しいなと感じました。
そこで今回は理解できた部分だけ使い将棋AIを作ろうと考えました。なのでもちろん理想とするところはやねうら王のような偉大な探索部やNNUE-pytorchの中身を理解しつつ実装することなのですが、現状ではフルスクラッチの探索部と手書きの評価関数を組み合わせた、人間の自分と同格の芽生えたばかりのAIです。
本来このような状態で大会に参加するのは参加者への敬意に欠けているかもしれないと迷いましたが、将棋AIが面白い、やりたいという気持ちは本物なので、このアピール文章提出から大会までの一ヵ月の間にも成長させて勝負をできたらと思います。よろしくお願いします！

257　アピール文書

開発チーム

名前はまだない

開発環境・言語

Ubuntu・Python

257 とは

WCSC35 で初出場する強豪（になる予定）の将棋ソフトです。学生を中心としたチームで開発しておりディープラーニング系の将棋ソフトになっています。名前の由来は 2^8＋１で暗号化やハッシュ関数で用いられている最大値である 256bit よりもさらに大きな数字を目指すという点から命名しました。ベースとなるコードは DLShogi を用いて開発しています。参考文献として「強い将棋ソフトの作り方　～Python で実装するディープラーニング将棋 AI～」（著）山岡忠雄/加納邦彦を用いています。

棋風

内部定跡を一切開発しておらず、初手からよく言えば定跡にとらわれない力戦的な将棋を指します。角換わりや相掛かりの定跡開発が進んでいる将棋 AI の大会で力戦の棋譜を楽しんでいただきたいと思います。

技術的な工夫

1.入力特徴量の変更

改造点としては入力特徴量の変更が挙げられます。DLShogi では駒の種類と配置、手駒を入力特徴量としていましたが、我々は駒の利きやヒモ、囲いの状態などの特徴量を与えて実験を行いました。駒の配置に基づくものでは NL で読み替えが効くものの、その情報をより強く結果に意識させることができました。大会当日までに一部の情報を薄めたり、他の観点を組み込みたいと考えています。処理時間は...がんばります....。

2.活性化関数、誤差関数の比較実験・変更

ニューラルネットワークに用いられる活性化関数にはシグモイド関数、Tanh、ReLU、ELU などがあり、新しい関数が次々に開発されています。また誤差関数も交差エントロピーや平均二乗誤差など複数の手法が開発されています。これらの関数を用いてモデル開発を行い、比較実験を繰り返し行うことでより将棋 AI に適した活性化関数、誤差関数を検討します。

技術的な工夫というよりは、来年度に向けてのデータ収集の意味合いのほうが強い取り組みになります。

今後の展望

棋譜を用いた教師あり学習と自己対局による学習を繰り返し、強化を行います。また他分野の AI に用いられている技術などの学習も積極的に行い、取り入れられる技術はどんどん取り入れていこうと思います。

大会に向けて

若手らしく積極的な取り組みを披露していければと思います。よろしくお願いいたします。

参考文献

「強い将棋ソフトの作り方　～Python で実装するディープラーニング将棋 AI～」（著）山岡忠雄/加納邦彦