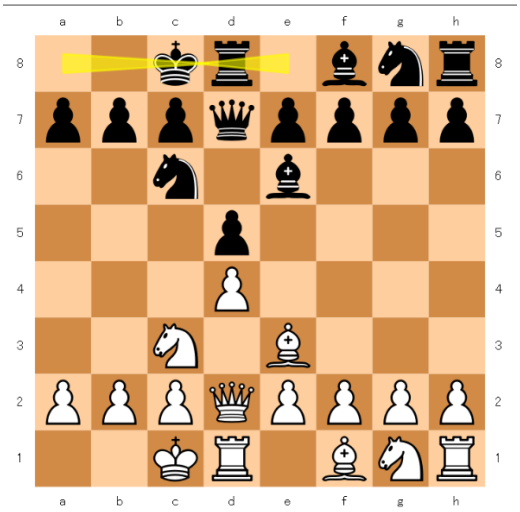
HoneyfishchessAI プロジェクト設計資料

第 1 章：はじめに

プロジェクト概要:

1. 特定の戦略（クイーンサイド・キャスリング以後 O-O-O）に有利な局面を識別する特化型 CNN (SE-ResNet）の設計と学習。
2. 学習済みモデルの UCI 準拠チェスエンジンへの統合。
3. 単純な Top-K 探索（v0.2）と伝統的なアルファベータ探索＋反復的実装を行った（v0.3）という 2 つの異なるアーキテクチャの比較分析。
4. ML モデル（v3PT、v2_latestPT）がエンジンの指し手選択に与える実践的影響の評価。



- **実施内容**: AI 作成未経験であったが、PyTorch を用いて、チェスにおける特定の戦略（O-O-O）の実施判定を、二値分類を評価として機能する特化型 CNN（SE-ResNet）を設計・訓練した。次に、このモデルを UCI 準拠のエンジンに統合し、SE‐RESNET＋TOP－K探索エンジンと予備設計で、SE‐RESNET＋伝統的な α-β 探索エンジン（$\alpha\beta$ コア＋静止探索・指し手順序付け・反復深化・TT）での指し手選択への影響を検証した。主要な発見は 2 点ある。(1) 訓練データにおけるターゲットの指し手の極端な不均衡は、高い AUC スコアが F1 スコアで測定される実世界性能の低さを覆い隠しうるという、古典的な機械学習の課題を提示した。(2) 対戦結果は、学習モデルに対して、エンジンの探索を元にしたアルゴリズムの優位性を実証し、ゲーム AI の基本アーキテクチャに関する実践的なケーススタディを提供した。
- **本資料で示すこと**:O－O－O判定 2 値分類　におけるデータ収集、データラベリング、学習モデルの作成、混同行列の具体的な実施やデータの見方。
  3 つの異なるアプローチの実戦、エンジン（v0.2 と v0.3）と学習モデル（v3PT と v2_latestPT）による比較の過程と、得られた知見。

第 2 章：課題設定と初期アプローチ（v0.2）

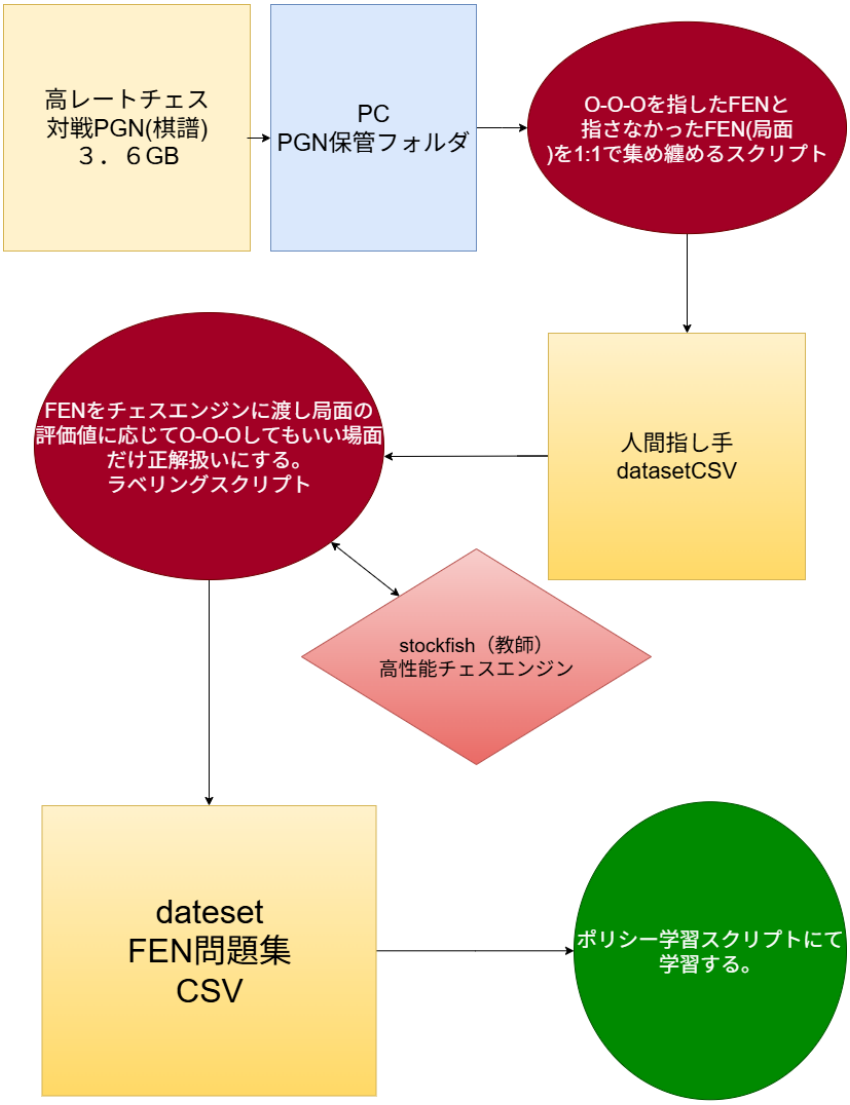
課題：将棋チェスには様々なＡＩが存在する。しかし特定の戦術に沿ったＡＩや機械学習の存在は限定的である。（振り飛車 AI honeywaffle など）この為、まずは UCI エンジンと学習モデルとの統合し、GUI で動かす事を目標とする。

仮説：Ｏ－Ｏ－Ｏのエンジンや学習モデルでの評価値を高くなるように調整、競合の手であるＯ－Ｏをコードで禁止にしたら指すと予測。

v0.2 のアーキテクチャ:

- **ルールベース:** コードで O-O を強制的に禁止する。
- **ポリシーエンジン:** 、指し手を 1 手先のみ SE - RESNET＋TOP－K探索エンジン検索。Ｏ－Ｏ－Ｏに対して崩れる手は-0.06、O-O-O には 0.05 評価値を上げるなどの補正を掛ける。また学習モデルの内容も反映させる。その中から最も良い手、選ぶ。

第 3 章：データ準備とモデル学習



前項のフロー図に基づいてデータセットを作成する。今回は高品質な O-O-O を反映したデータを学習させ O-O-O の判定に用いるため、レート 2200 以上のチェスプレイヤー同士の対局棋譜をオンラインより DL。その後データセットスクリプトにて O-O-O が合法局面にて、O-O-O を指した局面を陽性データ、指さなかったデータを陰性データとして、O-O-O 直前の局面を表す FEN データを datasetCSV に集める。

その後 CSV に保存されている FEN のラベリングを行うが、教師あり学習において正解の一貫性を担保する為、現在最強チェス AI と呼ばれている stockfish を活用する。こちらをラベリングスクリプトから、CP 設定のコマンドに応じて、O-O-O を指して良い局面なら陽性、他の手が良い局面なら陰性と言うような形で FEN データに陽性陰性をラベリング。datasetFEN 問題集 CSV を作成後。学習モデルにこちらを学習させる。

モデルの学習と評価:

モデル概要: Fen, move, label を読み込み、層化で train(学習データ)/val（検証データ）に分割。

FEN データを盤面 20ch に分ける。指し手は 2ch

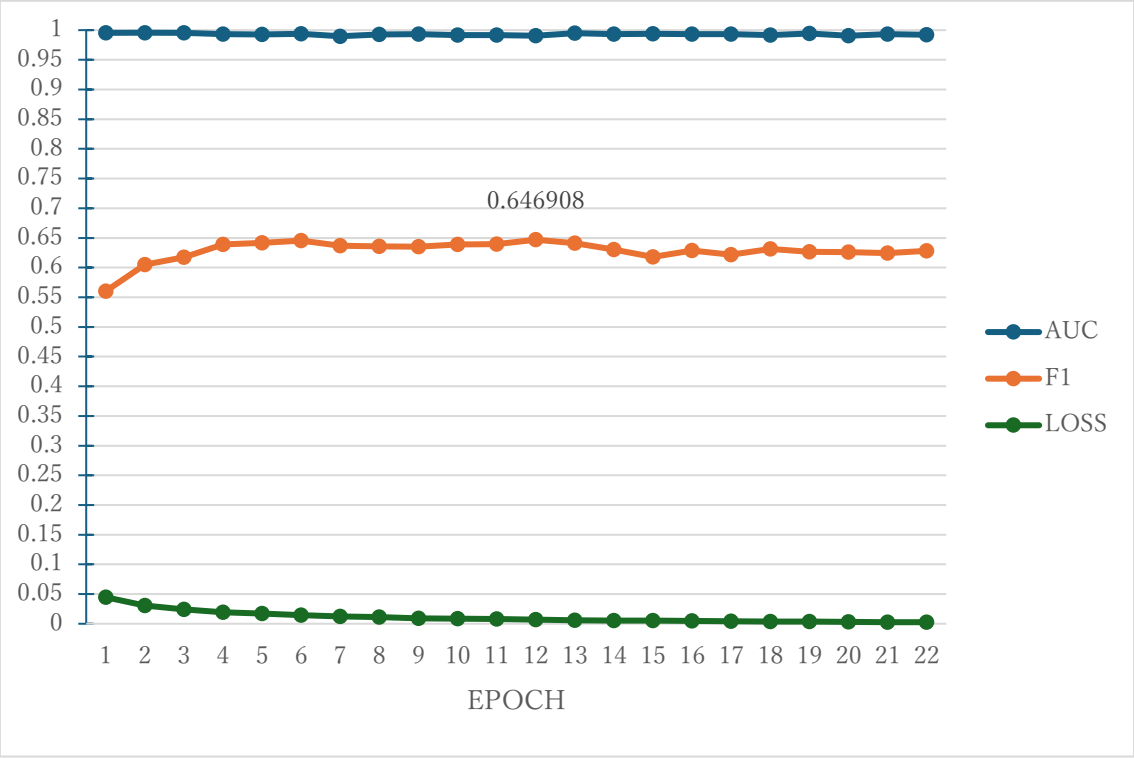
これらを結合し、テンソル化し、モデル CNN（小型 SE-ResNet）にて、盤面が O-O-O するべき盤面か、しない方がいい盤面かで学習。

BCEWithLogitsLoss＋WeightedRandomSampler を用いて問題集データの陽性陰性不均衡対策を行う。

しきい値＝0.5 にて実施する。

評価

EPOCH	LOSS	ACC	PRECISION	RECALL	F1	AUC
1	0.044582	0.985417	0.391772	0.982642	0.560198	0.995614
2	0.030721	0.988274	0.443768	0.949373	0.604823	0.995659
3	0.024052	0.989241	0.464991	0.918997	0.617528	0.995182
4	0.01954	0.991059	0.516677	0.836548	0.638807	0.993113
5	0.017029	0.991492	0.533035	0.805207	0.641444	0.992884
6	0.014216	0.991437	0.530251	0.824012	0.645271	0.99391
7	0.012366	0.991496	0.533943	0.788814	0.636824	0.989707
8	0.011087	0.990922	0.51208	0.837994	0.635699	0.992798
9	0.009139	0.991419	0.530936	0.79026	0.635148	0.993162
10	0.008356	0.992225	0.569329	0.728544	0.639171	0.991904
11	0.007952	0.992476	0.584499	0.7054	0.639283	0.991575
12	0.007041	0.991856	0.548042	0.789296	0.646908	0.990791
13	0.006086	0.991975	0.555242	0.758438	0.641125	0.994992
14	0.0055	0.992143	0.56762	0.708293	0.630202	0.993242
15	0.005462	0.992841	0.623466	0.612343	0.617855	0.993912
16	0.004835	0.992303	0.577778	0.689489	0.62871	0.993212
17	0.004333	0.991948	0.559061	0.700579	0.62187	0.993128
18	0.003833	0.992116	0.565699	0.714079	0.631287	0.991588
19	0.003703	0.992244	0.57494	0.688042	0.626427	0.994107
20	0.003201	0.992294	0.578195	0.682739	0.626133	0.990716
21	0.002854	0.992112	0.5676	0.694311	0.624593	0.993143
22	0.002586	0.992262	0.575502	0.690935	0.627958	0.991929



いずれも v2_latestPT の混同行列の数値、AUC、LOSS、F1 に着目し、折れ線グラフで表示する。EPOCH12 にて F1 が頭打ちとなり、それ以上は伸びず、自動停止機能により、22 回で終了している。F1 が 0.646 であったのが気になったが、AUC や ACC の数値も 90 後半超えなので、モデル学習は正常であると考察する。

第 4 章：v0.2 の実験結果と新たな課題

GUI に v0.2UCI、v2_latestPT 統合した CMD を登録。
UCI の修正を繰り返して、動作確認後、O-O-O を実施するが、レートが高くない事が考察される。また自分自身で何百局と指して検証することも難しく、他の UCI エンジンを対戦で用いるも、 v 0.2 はレートが低く対戦で検証が難しいと言う結論に至った。

上記の結果から、新たな課題として

学習モデルがどのくらい O-O-O を指すことに影響を与えているのか。CMD で O-O 禁止にしている状態で、AI に O-O-O を選択させていない。v0.2 はノータイムで思考時間無く指し、探索エンジンや O-O-O に寄せた規則ばかりでレートが低い。今まで O-O-O を指す動く AI を目指していたが、AI の指し手が O－O－O だけでなく、可能な限り攻めの 1 手を目指し、レートが高いエンジンの作成も課題で有ることに気づく。

またラベリングスクリプトと学習モデルのコードでO－O－Oに関するルールの見落としがあり、共通の修正箇所があり。

5 章発展的アプローチ（UCIv0.3、v3PT）

今後の検証相手には UCIv0.2 に反復的な機能追加、修正を行い UCIv0.3 とエンジンの予

備設計を行った。以下の表に変更内容を示す。

項目	v0.2	v0.3
意思決定の核	policy Top-K（1プライ）	**α β**（Negamax）＋反復深化
葉の安定化	なし（簡易ペナのみ）	**静止探索**（取り/昇格延長）
指し手順序付け	policy降順＋軽ヒューリ	**policy降順＋取り/王手優先**（全ノード）
置換表	なし	**あり**（深さ・値・手）
時間思考	なし（固定遅延想定）	**あり**（movetime/残時間/増加）＋Timeout
モデル/CNN	同一構成	**同一構成**（任意で0-0-0即合法フラグ面）

v0.3 の改良アーキテクチャ:

v0.2 は、CNN のポリシーに従う単純な 1 手読みエンジンで、目標とする挙動を誘発できる。結果: レート低い事が予測され、有効なテストが困難。アーキテクチャの根本的な欠陥が露呈。

v0.3 は、ML がモデルの影響を正しく評価するには、安定したベースラインとして堅牢なアルファベータ探索フレームワークが必要。結果予想：大幅に強化されたエンジンが完成し、意味のある比較分析が可能になった。

またコード修正後再度**人間指し手 datesetCSV（フロー図参照）**からの手順に基づいてラベリング＋学習を行い v3PT を生成する。

第 6 章：最終実験と考察

バージョンの違い	勝ち	引き分け	対局数	0-0回数	0-0-0回数
hoenyfishUCI	1	379	400	173	12
hoenyfishUCI2	20	379	400	3	1
Elo difference: -16.5 +/- 17.4, LOS: 17.1 %					
hoenyfishUCI2	14	279	300	0	1
hoenyfishUCI2V2PT	7	279	300	0	0
Elo difference: +8.1 +/- 20.1, LOS: 65.7 %					

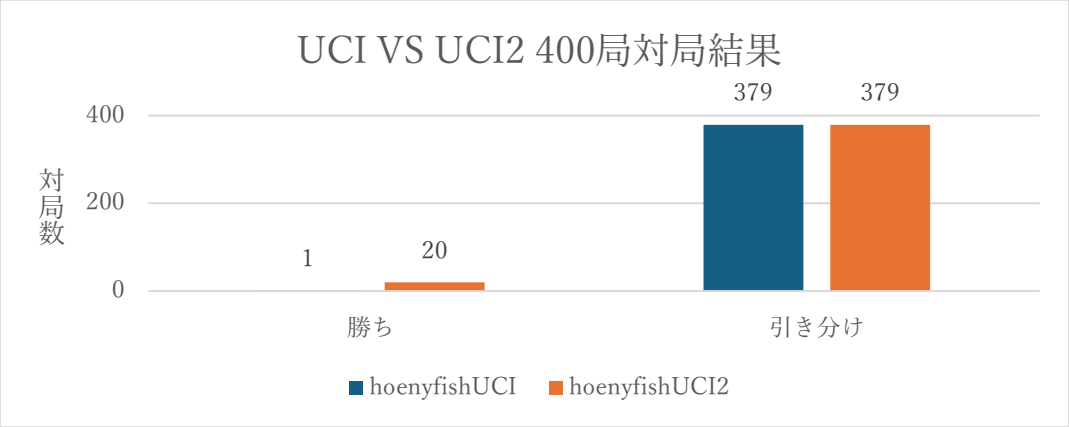
UCI 同士で対戦。上記表は対戦結果やキャスリング回数、レートやそのレートの妥当性を表示するデータであるまた、統合エンジンの名称と実際のエンジンとモデルのバージョンの組み合わせなどは下記の太文字の名称で記載。

honeyfishUCI＝UCIv0.2+ v2_latestPT 統合エンジン

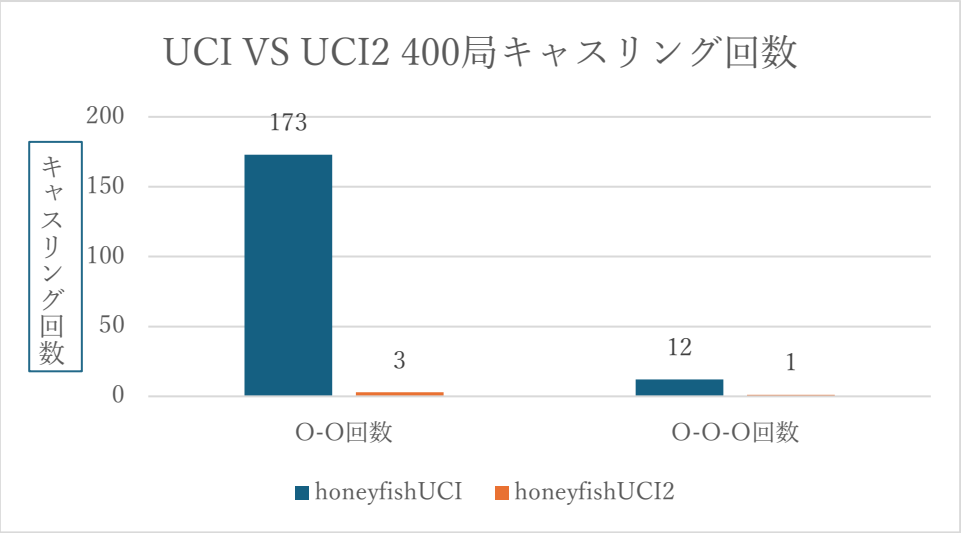
honeyfishUCI2=UCIv0.3+v3PT 統合エンジン

honeyfishUCI2V2PT＝UCIv0.3+ v2_latestPT 統合エンジン

エンジン同士の条件を整える為、honeyfishUCI2(**以後 UCI2**)の方はノータイムで指す honeyfishUCI(**以後 UCI**)エンジンに対して時間切れを起こさせないように 0.02 秒で指すように設定、置換表の深さも 1 に変更し、お互い GPU で手を読ませる方向に、また結果が偏らないように、UCI から O-O 禁止のコード削除。以下対局結果とキャスリング回数のグラフを作成し、結果を纏める。



400 回対戦して、引き分けが圧倒的に多い形になりましたが、勝ち自体は UCI が 1 に対して、UCI2 が 20 勝しており、レートも 16.5 程度↑であり、UCI の方が loss(レートが高い確率)も 20%を割っている。



キャスリング回数の詳細を見ると、UCI は 40%、UCI2 は 1%程度しかも、O-O-O 選択の確率を見ると、**UCI でも 3%、UCI2 に至っては 0.25%**

これら結果から UCIv0.2 は O-O 禁止のコードを抜いたから、O-O-O 選択の確率が下がることは予測でき、その様な結果になった。UCIv0.3 は短時間での指し手でもレートは向上しているが、そもそもキャスリングを行わない傾向があるのではないかと仮説が浮上した。またモデルによる O-O-O の選択が優先されるか否かに関しても、O-O-O 選択はされず、モデルよりもエンジンの探索アルゴリズムやルールが指し手の方が反映されやすいのでは、と言う仮説が浮上した。

急遽 honeyfishUCI2V2PT（**以後 UCI2V2PT**）を作成、モデルの違いで O-O-O の差がでるのか、またエンジンによってキャスリングに影響するのかを 300 局対戦し、キャスリングは UCI2 が 1 回 O-O-O を行ったのみであった。

考察

UCI が v0.2 から v0.3 変更後、キャスリング全体を指さなくなった理由に関しては、$\alpha\beta$ 探索の順序付けが取り/王手先行されおり、キャスリングより優先されたと考えられる。

また O-O-O を指さない理由に関して、学習モデルからの O-O-O 誘導を狙って、指し手のボーナスに関して、v0.2 からの変更で O-O-O の権利を放棄するペナルティや、指したときのボーナスを削除。これらの事から O-O-O を指される確率が減った要因である。

最後にラベリング後の FENCSV の著しい不均衡と混合行列の表示について。3 章での評価にて、F1 と ACC と AUC の評価を見て、F1 の数字の 6 割台であり、正常と判断。しかし Recall、Precision の擬陽性、偽陰性の判定の評価をせず。改めて評価を行い、ラベリング後の FENCSV の情報を確認すると、陽性 1 に対して、陰性 80 と不均衡な割合であった。この為、しきい値を下げる。現在不均衡割合の調整は Weight などで現状 1:1 にて行っている。こちらを適切な割合、1:3or1:5 にして、F1 率を上げる方向で調整を行うと、より O-O-O を指せるモデルになったのではないかと考えられる。

この為モデル PT が正しく機能していなかった可能性も考えられるが、そもそも v0.2 でも O-O 禁止のルール撤回後、評価関数やヒューリスティクス的なボーナスとモデルを使っていたが、O-O-O を 3%程度しか、指されなかった結果も出ている。

これらの事から学習モデルよりも、探索エンジンの評価が、指し手に強い影響を及ぼす。

第 7 章：結論と今後の展望

結論: **アルゴリズムの優位性**: 実験結果は、本ハイブリッドアーキテクチャにおいて、指し手選択の主要因は探索アルゴリズムであり、統合された CNN の評価値の影響は限定的であることを明確に示している。

評価指標の重要性: 本研究は、クラス不均衡データに直面した際の適切な評価指標選択の重要性を強調する。AUC スコアよりも F1 スコアが、モデルの実用性能をはるかに正確に反映する指標である。

アーキテクチャのトレードオフ: v0.3 における駒取りや王手を優先する指し手順序付けの変更は、Elo レート向上に寄与した一方で、意図せず著しいキャスリングの優先度を低下させた。これはチェスエンジン設計における複雑な相互作用とトレードオフを浮き彫りにしている。

今後の展望:

本プロジェクトでは、評価関数を差し替えるアプローチを試みたが、訓練データの不均衡という課題に直面し、AI の指し手を意図通りに制御するには限界があることが示唆された。

この結果を踏まえ、次のステップとして、より根本的なアプローチの変更を検討する。具体的には、従来の $\alpha\beta$ 探索から**モンテカルロ木探索（MCTS）**のような代替アルゴ

リズムへ移行する。さらに、自己対戦による強化学習を導入し、AI 自身が試行錯誤の中で特定の戦術の有効性を学習し、自身のレートを高めるモデルを構築することを目指したい。