Ryfamate Cross Network

Komafont*

2023 年 4 月 21 日

1 はじめに

Ryfamate は、従来よりコンピュータ将棋の発展に寄与してきた NNUE 型評価関数 [1] と、近年目覚ましい成長を遂げる Deep Learning (DL) 系評価関数 [2] を組み合わせ、それぞれの良さを活かすことを目標としている。2021 年には NNUE 型評価関数と DL 系評価関数の変則合議を採用し、2022 年には変則合議に用いる DL 系評価関数に、自然言語処理の分野で注目される Transformer 型の評価関数 [3] [4] を導入した。2023 年の今回は、DL 系評価関数の主流である ResNet に、NNUE や Transformer の持つ性質を取り入れた、新しいアーキテクチャ Ryfamate Cross Network (RyfcNet) を採用する予定である。

本書では、日本語と図を用いて簡潔に RyfcNet を紹介する。

2 背景

現在、コンピュータ将棋に用いられる DL 系評価関数は、画像認識において高い性能を有する深層畳み込みニューラルネットワーク、ResNet [5] である。この畳み込みは、盤面のうち主に 3×3 の部分空間ごとに特徴量を得るものであるが、その性質上、離れた位置にある駒の関係を認識するためには多数の畳み込みを行う必要がある。また、この畳み込みにおいては、位置によらず同じ重みパラメータが適用されるため、駒の絶対座標の情報を学習に生かすことが困難である。これらの問題を解決するため、RyfcNet では、既存の ResNet に、次に紹介する新しい層を導入する。

* 駒の書体 (Komafont) https://twitter.com/komafont

3　アーキテクチャ

RyfcNet では、畳み込みを、任意の次元について入力空間と同じ長さを持つカーネルを、それ以外の次元の方向に移動させながら適用する変換と捉え、その次元を層ごとに適切に選択することで、入力空間上の離れた位置にある情報の関係を少ない回数の演算で効率的に認識する。具体的には、ブロック数やチャンネル数などに応じ、次の層を組み合わせて構成される。

(1) S-Layer

通常の畳み込み層であり、チャンネル方向に入力空間と同じ長さのカーネルを持ち、縦横方向にカーネルを移動させながら適用する。

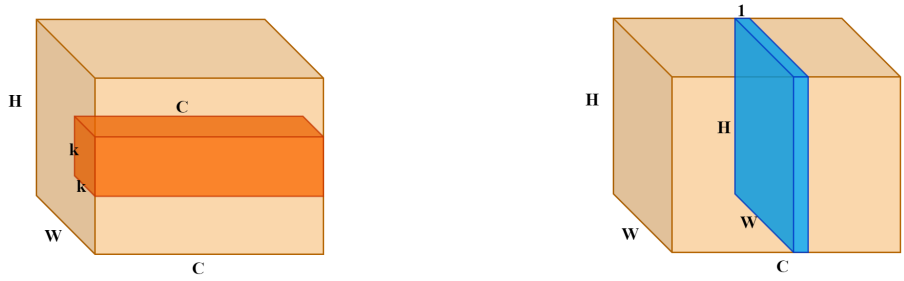
(2) F-Layer (**図** 1)

チャンネルごとの全結合または空間方向 (縦横方向) の self-attention を行う。これによって、盤面全体の情報を、少ない回数の変換で認識することができる。ただしこの変換は、チャンネル数によっては通常の畳み込みと比べて著しく推論速度が遅くなるため、S-Layer と組み合わせてチャンネル数を調整することが有効である。

(3) C-Layer (**図** 2)

任意の 2 つ (以上) の次元について入力空間と同じ長さを持つカーネルを、それ以外の次元についてのみ移動させながら適用する畳み込みを行う。盤面が 9×9 の将棋の場合、通常の畳み込み層は 3×3 の部分空間ごとの計算を 81 回行うのに対し、この畳み込み層では 9×1 や 1×9 の部分空間ごとの計算を 9 回行う。これにより、通常の畳み込み層と同数のパラメータを持ちながら、少ない演算回数で、格子方向に離れた位置にある駒の関係を認識することができる。例えば、図 3 のように、離れた位置にある飛車や香の間接的な効きを少ない回数の畳み込みで認識することができる。さらに、この畳み込み層は選択した次元に移動しないため、重みパラメータは位置に依存したものとなり、駒の絶対座標の情報を学習に生かすことが可能である。これにより、例えば、駒が特定の行 (段) や列 (筋) にあるときのみ反応する畳み込みを行うことができる。

上記の層に加え、ネットワークの構成に応じて、畳み込み層への入力に対し、絶対座標に依存する学習可能パラメータを付与する possition embedding を行う。これにより、例えば自陣や敵陣でのみ反応する畳み込みや、角や桂が初期配置から移動できる位置でのみ反応する畳み込みを行うことができる。



(a) S-Layer　　**(b)** F-Layer

図 1: S-Layer と F-Layer
S-Layer がチャンネル方向に多くの情報を伝えるのに対し、F-Layer は空間方向 (縦横方向) に多くの情報を伝える。

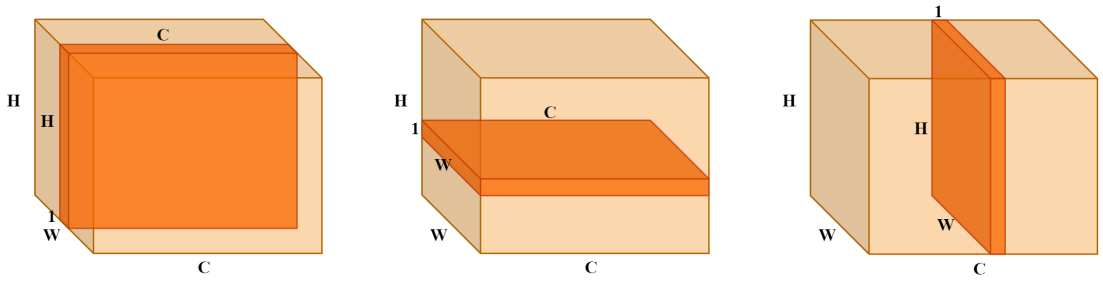
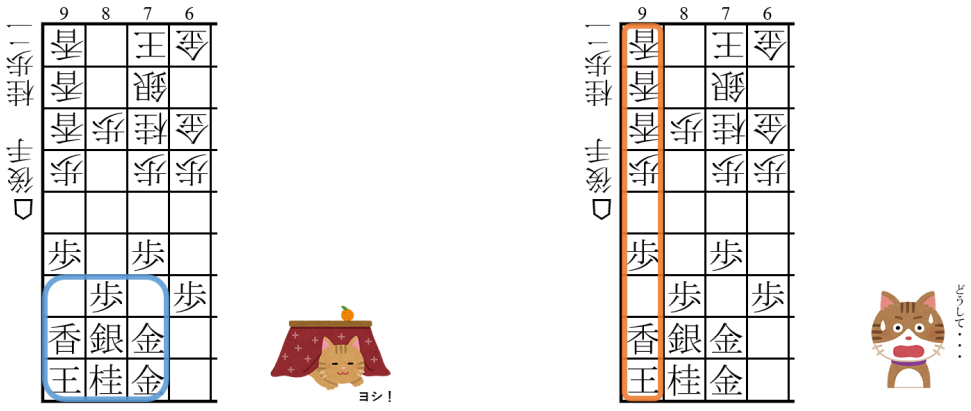
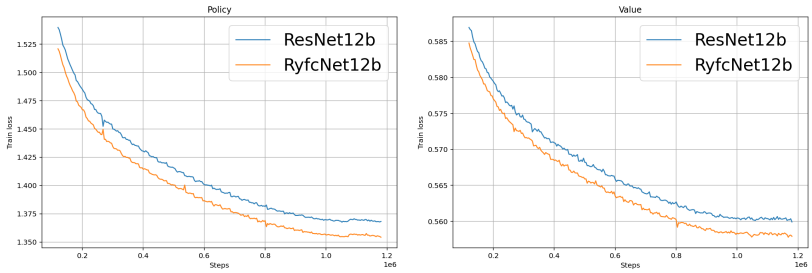


図 2: C-Layer
複数の種類の C-Layer を組み合わせることで、離れた位置にある情報を効率的に伝えることができる。



(a) S-Layer　　**(b)** C-Layer

図 3: S-Layer と C-Layer
S-Layer では離れた位置にある駒の影響を認識するためには多数の畳み込みを行う必要があるが、C-Layer では飛車や香の間接的な駒の効きなど格子方向に離れた位置にある駒の関係を少ない回数の畳み込みで認識することができる。



	Res12b	Ryfc12b
Test Policy Loss	1.400	1.385
Acc.	53.49%	53.74%
Test Value Loss	0.581	0.579
Acc.	66.81%	66.87%
NPS	34937	37271

図 4: 学習結果

dlshogi [6] を用いて、2.4 億局面の教師 (hcpe3 形式) を作成し、10 epoch 学習を行った。バッチサイズ 2048。学習率は、Warmup+Cosine Annealing によって調整している。

テストデータは、floodgate および DL 系評価関数と NNUE 型評価関数による自己対局の棋譜を用い、評価値から推測される期待勝率が 99.995% 以内の 512604 局面をサンプリングしたものを用いた。推論速度 (NPS) は、世界コンピュータ将棋選手権および世界将棋 AI 電竜戦にて現れた 5 局面を 10 秒間推論させ、平均値を求めた。

4 実験結果

既存の ResNet と本書の RyfcNet を比較するため、同一条件下で ResNet 12 ブロック 192 チャンネル (Res12b) と、RyfcNet 12 ブロック 192 チャンネル (Ryfc12b) を学習した。その性能を比較したところ、図 4 のとおり、Ryfc12b は精度・推論速度ともに Res12b を上回り、対局の結果、表 1 のとおり、 +48.1 (±19.8) のレーティングを記録した。[*1]

#	PLAYER	:	RATING	ERROR	PLAYED	(%)	CFS(%)	W	D	L
1	dr2_exhi_600ms_140	:	49.7	25.9	800	53.6	55	411	36	353
2	Ryfc12b_ep010_1000ms_140	:	48.1	19.8	1600	56.8	100	870	76	654
3	Res12b_ep010_1000ms_140	:	0.0	----	1600	46.6	100	706	80	814
4	YO763_S5_SC24_16t850ms	:	-50.3	26.1	800	39.6	---	300	34	466

表 1: 対局結果

cshogi のリーグ戦機能 [7] を用い、Res12b、Ryfc12b、dr2_exhi [8] による 3 者リーグと、Res12b、Ryfc12b、水匠 5 [9] による 3 者リーグをそれぞれ 1200 局、合計 2400 局実施した。Res12b、Ryfc12b の思考時間は 1 手 1 秒とし、それとおおむね互角の強さとなるよう dr2_exhi と水匠 5 の思考時間を調整した。Res12b、Ryfc12b の探索パラメータは、dr2_exhi のデフォルト値を用いた。初期局面は、36 手目の互角局面集を作成し用いた。

レーティングは Ordo [10] を用いて計算している。

[*1] 実験環境：
Ryzen Threadripper 2950X, GeForce RTX 3090, WSL2+Docker(nvidia/pytorch:22.12)

5 おわりに

本書では、比較実験のために 12 ブロックのモデルを用いたが、大会では、15 ブロックまたは 20 ブロックのモデルを用いる予定である。比較実験においては、自作の教師データのみを用いて学習したが、大会用のモデルでは、自作の教師データに加え、加納氏・山岡氏の公開した教師データ [11] から約 6 億局面、たややん氏が公開した教師データ [12] から約 1.2 億局面を利用している。また、探索部は、dlshogi とやねうら王 [13] を一部改良して用いているほか、教師局面の作成や計測には floodgate の棋譜や公開された多くの評価関数を利用している。限られた時間と計算資源の中で新しいモデルアーキテクチャを開発するには、これらの膨大なオープンソースが必要不可欠であり、ここに感謝の意を表したい。

なお、本書で紹介した新しいモデルは、ご協力いただける方がいれば学習を進めたうえで公開したいと考えており、多くの方にご賛同いただければ幸いである。

参考文献

[1] 那須悠. 高速に差分計算可能なニューラルネットワーク型将棋評価関数. 2018.

[2] 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.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, Vol. 30, , 2017.

[4] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. arXiv preprint arXiv:2006.03677, 2020.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[6] `https://github.com/TadaoYamaoka/DeepLearningShogi`.

[7] `https://tadaoyamaoka.hatenablog.com/entry/2021/04/06/225615`.

[8] `https://tadaoyamaoka.hatenablog.com/entry/2021/08/17/000710`.

[9] https://twitter.com/tayayan_ts/status/1463107309492531202.

[10] https://github.com/michiguel/Ordo.

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

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

[13] https://github.com/yaneurao/YaneuraOu.