やねうら王 featuring Ryfamate
WCSC34 大会後アピール文

チーム やねこま王

やねうらお [*], 駒の書体 (Komafont) [†], よみぃ [‡]

1 開発動機

筆者 (Komafont) は、これまで NNUE 型評価関数[*1] と Deep Learning (DL) 系評価関数を組み合わせた合議プログラム Ryfamate として単独出場していたが、DL の発展に伴い大規模な計算資源の必要性が増していることや、持病のため現地参加を求める新ルールへの対応が困難であったことから、単独出場を断念しチームで参加することとなった。そこで、Ryfamate の合議に用いる独自開発した評価関数 Ryfamate Cross Network (RyfcNet) [*2] のうち、序中盤向けに学習を進めていた 30 ブロック Attention [*3]付きのモデルをチームに提供した。これに、やねうらお氏が作成したペタショック定跡[*4] および探索エンジン ふかうら王[*5] を組み合わせ、よみぃ氏が担当したインフラのもとでさらに学習を進めたうえで出場した。

2 開発過程

近年、生成 AI などの分野で目覚ましい成果をあげる Transformer であるが、そのまま将棋用の評価関数とした場合は推論速度が問題となる。そこで、RyfcNet では、現在将棋の評価関数として活躍している ResNet [*6] の一部ブロックを、Transfomer の中核的要素である Attention を用いたブロック (A-Block) に置き換え、従来のブロックと同程度の推論速度になるよう調整した。これにより、12 ブロックの比較実験において、Attention を用いない RyfcNet と比べ、わずかながら精度の向上を確認した。加えて、Attention を用いた RyfcNet のほうが、チャンネル数を増加させたときに精度が大きく向上することなどから、序中盤向けの大型モデルに Attention を採用した。

さらに、本年は Positional Encoding (PE) を工夫し、Relative Position Representations [*7] などを参考に、Attention 中の各 Softmax 関数への入力値に対して学習可能パラメータを加算した。従来の ResNet では、角の進む方向に離れたマスの認識に複数のブロックが必要であったが、この工夫により、これを 1 つのブロックで認識できるような head が学習の結果として獲得できた。[*8]

---

[*] やねうらお https://yaneuraou.yaneu.com/

[†] 駒の書体 (Komafont) https://x.com/komafont

[‡] よみぃ https://www.youtube.com/@WoooRen1006

[*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] https://www.apply.computer-shogi.org/wcsc33/appeal/Ryfamate/appeal_ryfamate_20230421.pdf

[*3] Attention : 本文では Multi-Head Self-Attention のことを指す。

[*4] https://yaneuraou.yaneu.com/2024/01/14/the-era-of-large-scale-book-in-shogi-ai/

[*5] https://github.com/yaneurao/YaneuraOu

[*6] 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.

[*7] Shaw et al. Self-attention with relative position representations. 2018.

[*8] https://x.com/komafont/status/1787014338286653808

なお、Attention を用いたモデルでは、これを用いないモデルに比べ、早い段階で過学習の兆候が現れた。これは、Attention が Convolution に比べて帰納バイアスが弱く、同程度のモデルサイズでもより多くの教師が必要となるためと思われる。この点については、チームメイトから提供された計算資源によって大いに助けられた。また、今年は、たややん氏によって公開された NNUE 1000 万ノードの教師[*9] が特に終盤の Value の精度向上に貢献した。

3 実験結果

本年使用した Attention つきの RyfcNet(Ryfc30_WCSC34)、昨年合議のメインとして使用した Attention なしの RyfcNet(Ryfc20_WCSC33)、お前、 CSA 会員にならねーか？[*10] (tanuki-WCSC34) の三者を総当り式に対局させた。計測には、Ryzen 5950X + GeForce RTX 4070 ×1 のマシンを使用した。Ryfc30_WCSC34 と Ryfc20_WCSC33 は 1 手 4 秒とし、これらとほぼ互角になるよう tanuki-WCSC34 の秒数を設定した。

結果、Ryfc30_WCSC34 は、Ryfc20_WCSC33 に対して R+28.4 (±33.9, n=792) であった。[*11] 事前の実験により、長時間・高ノードの対局であれば優位性が拡大することは確認しているが、当初期待したほどの差ではなかった。Attention については、さらなる改良が必要であると考える。

4 追試可能性

すでに RyfcNet の基礎技術は再現可能な内容を公開しており、昨年公開して以来、複数の方より実装・実験したとの報告をいただいた。なお、C-Layer については、形状が (C,H,W) の入力値に対して、(C,H,1),(C,1,W),(1,H,W) といった形状の出力値となる。このため、形状が (C,H,W) のテンソルを得るためには、これらのうち複数の形状の出力値を加算することが有効である。[*12] また、A-Block については、PyTorch の標準的な方法で実装可能であるが、FP16 で学習や推論を行うとオーバーフローによって失敗することがある。このため、2024 年 5 月 5 日時点では、dlshogi の学習器・探索エンジンなどに、Attention 対応の修正を施す必要がある。当チームでは、オーバーフローの原因となる演算箇所を BF16 または FP32 で演算させることで対応した。

5 おわりに

大会主催者、スポンサー、関係者、そして開発者と開発者を応援してくださる多くの方々に、厚く御礼申し上げます。

[*9] https://x.com/tayayan_ts/status/1767133487847645457

[*10] https://lessertanuki.booth.pm/items/5713519

[*11] 実験の条件や結果の詳細は X(Twitter) で公開した。 https://x.com/komafont/status/1792004038596481499

[*12] 具体的な応用例として、2 種類の C-Layer の出力値を加算するネットワーク図を公開している。
また、C-Layer の出力値と S-Layer の出力値を加算することも有効と思われる。
https://x.com/komafont/status/1787011012945928504