1 Ray について

OSS 版の Ray に以下のような改良を施してあります.

1. 木探索の着手評価に Factorization Machines + Bradley-Terry モデルを利用
2. 木探索部とシミュレーション部に特徴を追加
3. 木探索部, シミュレーション部ともに細かなパラメータ調整
4. Deep Learning(後述)

2 Deep Learning 対応

2.1 利用する深層学習フレームワーク

今までは学習時に深層学習フレームワーク Caffe を利用していましたが, 今回からは PyTorch に乗り換えました. この変更により, 学習の高速化 (Automatic Mixed Precision の効果) や最新のネットワークアーキテクチャ, レイヤの実装を実現しました. LeelaZero や KataGo は TensorFlow を利用しているので、PyTorch を利用している人はあまり見ない気がします.

3 ネットワークアーキテクチャ

通常の Residual Network に以下のような手を加えています.

1. Batch Normalization を使わない代わりに SkipInit を利用.
2. Fully Connected Layer の代わりに Global Average Pooling を利用.
3. ReLU の代わりに FReLU を利用.
4. 各 Residual Block に SE モジュールを付与.

他にもいろいろ細かい工夫をしていますが, 内容がだいぶ細かくなってしまうので, ここでは割愛します.

1 つ目の工夫については, 主に SWA(Sthochastic Weight Averaging) の計算コストを小さく揃えることによる学習の高速化が狙いですが, レアな局面の特徴をうまく抽出できるようにすることも狙いです. CGF Open 2021 の 9 路盤で唯一敗けた Aya との対局でセキを正しく評価できていなかったのですが, いろいろ原因を探って見たところ, どうも Batch Normalization からの出力がいまいちになっていたようです.

2 つ目の工夫は KataGo でも使われているような感じで碁盤の大きさに依存せずに打てるようにすることが狙いです. 9 路では既に CGF Open 2021 で強化学習した Ray の生成棋譜 (約 300 万局) があるのでそれを流用し, 棋譜生成コストを削減します.

3 つ目, 4 つ目の工夫はネットワークの表現力を向上させ, 精度を向上させることが狙いです. 9 路で予備実験した時には SE モジュールの利用で +100elo, FReLU の利用で +80elo ほど強くなりました.

出力については以下の 5 つありますが, 対局時には Player's Policy, Value, Score の 3 つを利用します.

- Player's Policy : 次の自分の着手の予測
- Opponent' Policy : 自分が着手した後の相手の着手の予測

- Value：最終的な勝敗の予測
- Ownership：各交点の占有率
- Score：最終的な目数差の予測

3.1 自作 CUDA レイヤ

まともに性能測定をしていませんが, いろいろ工夫をした結果, cuDNN の API に対応するものが存在しない, または結構遅いレイヤがあったので, 以下のレイヤに関しては, CUDA で自作しました.

- FReLU Layer(Depthwise Convolution Layer)
- Concatenate Layer

3.2 強化学習の方式

AlphaGoZero 方式の強化学習をする予定です. 正確にはモンテカルロ・シミュレーションによる評価（プレイアウト）も併用するので, AlphaGoMaster に近いです. どこまで強くできるかはわかりませんが, 最後まで粘ってみようと思います.