DOI:10.18686/ahe.v8i6.13515

AI Game Go Based on Deep Learning

Jinyu Du, Dingli Wang, Xinming Li, Shuting Zhang, Zhigang Zhu *
City Institute,Dalian University of Technology, Dalian,Liaoning 116600

Abstract: With the rapid development of artificial intelligence technology, the game of Go, which is regarded as the peak of human intelligence, has gradually become a new hotspot of AI algorithm research. Based on deep learning technology, this paper selects convolutional neural network and residual network as model architecture, improves the traditional Monte Carlo tree search algorithm, proposes GoMCTS algorithm, introduces strategy value network to guide tree search, and comprehensively utilizes the data generated by human expert chess and self-game. Efficient distributed training through parallel computing and parametric server architecture.

Keywords: Deep learning; Go AI; Monte Carlo tree search; Computing resource optimization

Fund Project:
Innovation and Entrepreneurship Project for College Students in Liaoning Province (Project Number: X202413198087, Project name: Deep Learning-based AI Game Go)

Since the concept of artificial intelligence was proposed in the 1950s, Go has been regarded as one of the most difficult areas of mind games to conquer. Because of the large board, high branching factor and many changes in the game of Go, although the traditional algorithm based on game tree search has achieved great success in chess and other chess games, it is difficult to make a breakthrough in the field of Go. From AlphaGo defeating the European Go champion in 2015, to AlphaGo Master sweeping China's top players in 2017, to AlphaGo Zero that same year, which does not require human knowledge of chess and learns entirely through self-play, deep learning technology has shown amazing potential in the field of Go. It provides a new idea for building a highly intelligent Go AI system [1]. In this context, based on deep learning technology, this study aims to break through the bottleneck of traditional Go AI algorithms, and strengthen its explainability and practical value while improving the level of AI chess.

1. Model selection and architecture processing

In this study, convolutional neural network (CNN) is chosen as the main structure. Considering the similarity between the grid structure and the image of the Go board, CNN can well capture the local features of the board and their spatial relationships, so as to establish an efficient situation assessment model. The residual network (ResNet) is further introduced to solve the problem of gradient disappearance and degradation in deep networks. Combining CNN and ResNet, a 19-layer deep residual convolutional neural network is designed. The model takes a 19x19 Go board as input, extracts and converts features of multiple convolutional layers and residual blocks, and finally outputs the evaluation value of the current situation. BatchNorm regularization technique is used to optimize the model, which accelerates the convergence speed and improves the stability of the model. The loss function uses cross entropy loss and mean square error loss, and through a lot of experiments and tuning, the parameters of GoResNet-19 are finally determined.

2. Improved Monte Carlo Tree Search (MCTS) algorithm

Monte Carlo Tree Search (MCTS) is a heuristic search algorithm based on random sampling, which has been widely used in game tree search. The traditional MCTS algorithm consists of four main steps: selection, extension, simulation and backtracking [2]. On this basis, according to the characteristics of Go game, an improved MCTS algorithm GoMCTS is proposed in this study.

Firstly, the strategy value network based on deep neural network GoResNet-19 is introduced in the selection stage to guide the selection and expansion of tree nodes. The strategy value network fθ maps the checkerboard s to a probability distribution p and a scalar value v representing the probability and expected win in the current state, respectively, namely:

$$f_\theta(s)=(\mathbf{p},v),\quad \mathbf{p}\in\mathbb{R}^{19\times19+1},\quad v\in[-1,1]$$

When expanding new nodes, GoMCTS uses the improved PUCT (Predictor+UCT) algorithm to balance exploration and utilization based on the probability distribution p output by the policy value network, preferentially selecting child nodes with high probability and high confidence upper bounds:

$$a^*=\arg\max_a\left(Q(s,a)+\mathrm{c}_{\mathrm{puct}}P(s,a)\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}\right)$$

Where Q (s, a) is the action value of the node, P (s, a) is the prior probability, N (s, a) is the number of visits, and cpuct is the balance factor.

In the simulation phase, the traditional MCTS uses random games to evaluate the final game, while GoMCTS uses the win rate v output by the strategy value network fθ (s) to replace the simulation process and reduce the computational overhead. In order to deal with the huge complexity of the state space of the Go board, a double greedy strategy is introduced, which uses modified-UCT search inside the tree and a fast walk substrategy based on policy network sampling outside the tree to strike a balance between time and precision.

In the retrospective update phase, GoMCTS makes comprehensive use of the output of the strategy value network and the real game results to update the statistics of the tree nodes:

$$Q(s,a)\leftarrow\frac{Q(s,a)\cdot N(s,a)+v}{N(s,a)+1}$$

$$N(s,a)\leftarrow N(s,a)+1$$

GoMCTS also designed an adaptive search depth control mechanism, which dynamically adjusts the search depth according to the complexity of the current game, and obtains the best possible solution within the time limit.

3. Preparation and generation of training data

The data sources used in the study include human expert chess and self-game generated data.

For human expert chess data, professional match records are collected from major Go databases for cleaning and pre-processing. The pre-processing steps include: (1) Chess format conversion: converting chess files in different formats such as sgf and gib into a unified NumPy array representation; (2) Data enhancement: Through rotation, flipping and other transformations, each chess score is expanded into 8 equivalent state-action pairs; (3) Feature extraction: extract the characteristic plane of the chessboard state, including the distribution of pieces of one side and the other side, and the position of the previous step; (4) Label generation: According to the chess results and placement, the label of each step is generated, including the probability distribution of the move and the expected win rate.

For self-game generated data, use the trained strategy value network to self-play, so as to obtain a large number of high-quality chess data:

$$s_{t+1}\sim f_\theta(s_t),\quad a_t\sim\pi_\theta(\cdot\mid s_t),\quad z\sim\mathrm{sgn}(v_\theta(s_T))$$

Where st is the current state of the board, at is the move sampled according to strategy πθ, and z is the final result of the game. Through continuous self-game and parameter updating, the strategy network can continuously improve its chess level while generating data, thus forming a positive feedback loop [3].

4. Computational resource optimization and distributed training

In order to accelerate the training process and improve resource utilization, the optimization of computing resources was studied at the two levels of single-machine multi-GPU and multi-machine multi-GPU: (1) Single-machine single-CPU, the data parallelism and model parallelism strategies of TensorFlow were used to divide the training data into multiple sub-batches and perform forward and backpropagation computations on multiple Gpus in parallel; (2) Multi-machine multi-GPU, based on the parameter server architecture, design a distributed training system consisting of multiple parameter server nodes and computing nodes, and communicate through high-speed Internet. The parameter server maintains a global view of model parameters and responds to parameter pull and update requests from compute nodes, using an asynchronous update approach that allows compute

nodes to train independently at their own pace.

5. Experimental results and analysis

To comprehensively evaluate the performance of the Go AI system proposed in this study, two comparative experiments were designed:

(1) Several representative Go AI systems were selected as benchmarks, including Leela Zero, ELF OpenGo and Fine Art. Under the same hardware conditions, each AI system played 100 uncapped matches against this AI, and the results showed that the win rate of this AI against Leela Zero was 52%, with an average of +1.2 matches. 55% win rate against ELF OpenGo with an average of +1.7. The win rate against Juyi is 48%, with an average of -0.9 goals. Thanks to the improved MCTS algorithm and the optimization of the distributed computing architecture, the average decision time per step is 5.2 seconds, and the computational efficiency is similar to that of most comparison systems.

(2) Analysis of the result of playing with high-level human chess players. A number of professional Go players were invited to play a series of human-machine games with this AI. The game adopts Chinese rules, black paste 7 and a half, no seconds to read, each game is 2 hours. In 20 official matches, this AI has won 11 wins and 9 losses. Through the chess analysis, it was found that the AI showed high judgment in various stages such as layout, mid-table combat, and official competition, and could give high-quality decisions in some complex situations, but occasionally there would be judgment errors or tactical loopholes. During the game, the human players gave a positive evaluation to the overall performance of the AI, believing that it has the level of professional players, and also put forward valuable suggestions on local mistakes, providing a reference for the iterative optimization of the system.

6. Conclusion

This paper discusses the method of constructing a highly intelligent Go AI system by using deep learning technology. Through a series of innovations such as deep residual network GoResNet-19, improved Monte Carlo tree search algorithm GoMCTS, and search strategy guided by strategy value network, the level of Go AI's chess strength and computing efficiency have been effectively improved. Of course, the construction of general intelligence is still a long-term goal, and the existing GO AI does not have full human-like cognition and decision-making ability. With the further development of deep learning theory, the modeling methods and optimization techniques adopted in this research work can also provide references for other similar complex game and decision-making problems.

References:

[1] Zhang Sheng, Long Qiang, Kong Yinan. The Principle and method of AlphaGo series algorithms in Artificial Intelligence of Go [J]. Science and Technology Review, 2023,41 (07) : 79-97.

[2] Huang Jinhan. Research on Computer Go based on variable scale training and parallel Monte Carlo tree Search [D]. Guangxi University, 2023.

[3] Gao Tongtong, Ding Jiahui, Shu Wenao. Research on non-Go game System based on AlphaZero [J]. Intelligent Computer and Applications, 202, 12 (11) : 138-141+147.

About the author:

Jinyu Du, male, Han nationality, from 2004 to July, Tianjin, Dalian University of Technology, City College, undergraduate, software engineering major

* Corresponding author:

Zhigang Zhu, male, Han nationality, 1984-10, born in Shenyang, Liaoning Province, Associate professor title, teacher, Master degree, City College of Dalian University of Technology, research interests: Software engineering, Computer Science and Technology.