Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 13, MARCH 2023

DOI: 10.37943/13XCFG1746

Assem Dospanova
BSc student of Industrial Automation
Department of Intelligent Systems and Cybersecurity
201602@astanait.edu.kz, orcid.org/0000-0003-0218-722X
Astana IT University, Kazakhstan

Sanzhar Kusdavletov
Master of Science in Advanced Control and Systems Engineering,
Senior Lecturer
Department of Intelligent Systems and Cybersecurity
sanzhar.kusdavletov@astanait.edu.kz, orcid.org/0000-0003-0286-776X
Astana IT University, Kazakhstan

Aigerim Kalikova
MSc, Senior Lecturer
Department of Intelligent Systems and Cybersecurity
aigerim.kalikova@astanait.edu.kz, orcid.org/0000-0003-4429-0006
Astana IT University, Kazakhstan

AN EFFICIENT APPROACH FOR THE IMPLEMENTATION OF THE GOBANG GAME USING ARTIFICIAL INTELLIGENCE METHODS

Abstract. Gobang is one of the most ancient abstract strategy games for two players. The game is traditionally played on a board with black and white stones, where players take turns placing a colored stone on an empty intersection. The winner is the first player to form an unbroken chain of five stones, either horizontally, vertically, or diagonally. Although the rules of Gobang seem pretty straightforward, the game tree complexity is enormous since the board state is more intuitive than in other games. In this paper, we will implement an algorithm with faster and more efficient play than existing ones that will solve the Gobang game using artificial intelligence (AI) methods. The program will begin learning from scratch, then use self-play to produce training data, and eventually steadily build up its strength. The present work first focuses on the implementation of the supervised learning algorithm in the identification procedure in order to identify the position of the current fallen piece. This will be achieved by utilizing image processing and a residual neural network. Then a Gobang game procedure will be implemented using a game search algorithm, in which the state of the game is judged by means of a human-set function. After that, the function of judging the game state in the above game search algorithm will be changed to an artificial neural network (ANN) model since it is convenient to train a model with a small dataset. Finally, the reinforcement learning algorithm will be applied to learn the artificial neural network model so that the playing level of the Gobang game program can be continuously improved.

Keywords: Gobang, artificial intelligence, supervised learning algorithm, image processing, residual neural model, game search algorithm, artificial neural network, reinforcement learning.

Introduction

Artificial Intelligence (AI) is a broad-spectrum branch of computer science concerned with building agile machines capable of conducting tasks that typically require human intelligence.

DOI: 10.37943/13XCFG1746
© Assem Dospanova, Sanzhar Kusdavletov, Aigerim Kalikova

In the field of AI, games are a very important branch. For instance, in computer games, AI is used to engender responsive, malleable, or intelligent behaviors primarily in non-player characters similar to human-like intelligence. Two significant events: Deep Blue's triumph over chess champion Garry Kasparov in 1997 and AlphaGo's victory over go champion Lee Sedol in 2016, have brought computer games to the attention of people all across the world. The useful algorithms that Deep Blue and AlphaGo left behind are innovations in computer games and advance the science of artificial intelligence.

Previously, the Gobang game was concerned in the realm of computer games as a traditional two-player game with simple rules and an easy game process to imitate by computer. Computer programs were employed in 1994 and 2001 to demonstrate that the first hand will triumph over both the original forbidden hand and the original non-forbidden hand. The progress of computer Gobang, in contrast to computer chess and go, has been slower, and many Gobang specialists think that the top human players can still be surpassed by contemporary Gobang programs. Therefore, there is still a lot of interest in the development and innovation of different Gobang algorithms.

Related Works

The game theory-based minimax algorithm with the Alpha-Beta pruning tree algorithm was used to improve the entire Gobang game tree search process in [1]. Similarly, an integrated Gobang gaming application based on an intelligent algorithm has been designed and developed by employing the Alpha-Beta pruning tree algorithm [2]. A relatively dynamic evaluation strategy to optimize the Gobang evaluation function is studied in [3]. In order to increase CPU usage, an enhanced Gobang type evaluation method was suggested in [4]. A convolutional neural network with 32 convolutional layers and Monte Carlo tree search that constructs a self-learning intelligent Gobang algorithm is investigated in [5]. The study that focused on optimizing the establishment of a Gobang strategic value network to reduce computation and improve the training of the agent is presented in [6]. The approach presented in [7] proposed using decision trees for feature selection in self-play board game policies, presenting different decision tree types to reduce computational expense and improve interpretability.

Furthermore, in order to enhance the effectiveness of the evaluation function, deep learning algorithm was incorporated into the strategy creation and situation evaluation processes of the game decision-making mechanisms in [8]. A composite visual field network based on deep reinforcement learning that improves the network's thorough perception of both local and global chess shapes is proposed in the study [9]. The single conspiracy number (SCN) is utilized to identify different factors and analyze patterns that influenced a 5-in-a-row Gobang game [10]. The simulation test-bed uses a special Gobang dataset called «chess manuals» to reflect real-world game scenarios. The technique that involves Neural Network model using TensorFlow and Monte Carlo tree search algorithm is proposed in [11]. The work presented in [12] discussed the use of a distributed cluster structure for the development of a Gobang game server, which utilizes a thread pool and bitwise operations to improve operational efficiency and avoid lock competition.

In addition, a machine vision-based checkerboard replay approach is suggested in [13] to process the images and replay the gobang chessboard. This technique involves receiving images and converting them into the relevant array matrix. By configuring the checkerboard and reading the matrix of the checkerboard data from the images with MATLAB, the replay of the Gobang chessboard is visualized.

Board detection

The gobang board's features were identified using low- and mid-level computer vision techniques. The coordinates for the board's outside border and 361 individual squares were generated using those features. The main step in the process is implementing the Harris Corner Detector to generate intersecting horizontal and vertical lines. The intersections were grouped by distance using hierarchical clustering, and the groups' averages were used to get the final coordinates. Fig. 1 depicts the corner detection.

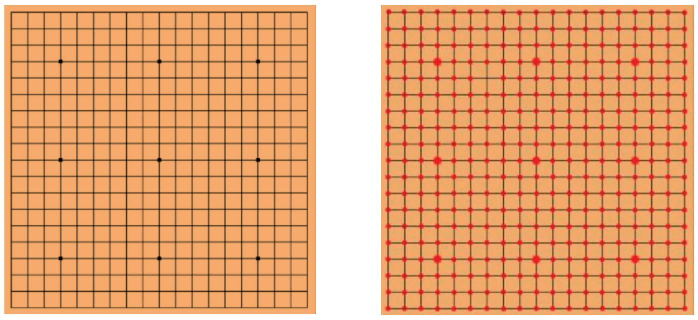

Figure 1. Gobang Board corner detector

In our context of the Gobang game, a residual neural network is used to classify the different pieces on the board. The input to the network would be an image of the board, which could be represented as a 19x19 matrix of zeros and ones, where zero represents an empty intersection, and one represents a stone of a particular color. The residual neural network would consist of a series of convolutional layers, which would extract features from the input image, followed by residual blocks that incorporate the shortcut connections. The final output layer of the network would consist of a set of neurons corresponding to the different types of pieces that need to be classified (e.g. black stone, white stone, empty intersection). The output of the network would be a probability distribution over these classes, which could be used to determine the most likely class for each intersection on the board.

Game algorithm

The Monte Carlo Tree Search (MCTS) game search algorithm [14] was employed to expand the scale of the game tree step by step through iteration. This algorithm mainly consists of four phases: selection, expansion, simulation, and backpropagation.

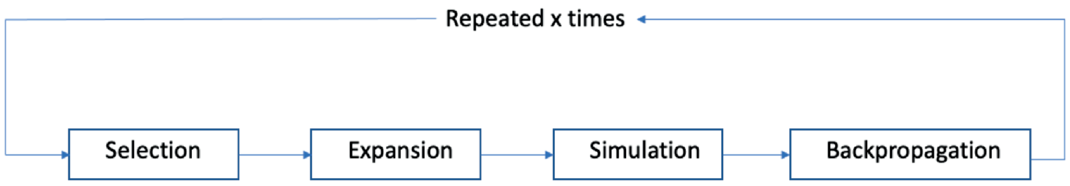

Figure 2. The design of the MCTS algorithm

The selection is based on the Upper Confidence Bounds (UCB1) formula:

$$\frac{W_i}{N_i} + \sqrt{\frac{C \times \ln N}{N_i}}$$

Here, the parameters of the equation are assumed as Wi is the number of times the child node wins, Ni is the number of times the child node participates in the simulation, N is the number of times the current node participates in the simulation and C is the weight coefficient.

Training Artificial Neural Network (ANN)

The Gobang game procedure does not require a huge dataset to train on, therefore it would be convenient to switch the function of judging the game state in the game search algorithm to an artificial neural network model. For now, it is essential to generate some input data to learn from. In this case, it is easy to create data by randomly choosing directions and observing if the pieces fulfill the game condition. On the input of our neural network, we will give an array of eight conditions:

- If there are any different pieces horizontally to the left of our piece (1→Yes, 0→No).
- If there are any different pieces horizontally to the right of our piece (1→Yes, 0→No).
- If there are any different pieces vertically above our piece (1→Yes, 0→No).
- If there are any different pieces vertically below our piece (1→Yes, 0→No).
- If there are any different pieces diagonally up to the left of our piece (1→Yes, 0→No).
- If there are any different pieces diagonally up to the right of our piece (1→Yes, 0→No).
- If there are any different pieces diagonally down to the left of our piece (1→Yes, 0→No).
- If there are any different pieces diagonally down to the right of our piece(1→Yes, 0→No).

By following the conditions listed above, we can create a sufficient amount of data to train a neural network to predict the game state in Gobang. Each piece placement on the board can be represented as a binary vector with eight values, where 1 indicates the presence of a different piece in that direction and 0 indicates the absence of a different piece. This data can then be encoded as a 1D vector x of length 361, where each element represents the state of a single intersection on the board. Each node in a hidden layer computes a weighted sum of the inputs and applies an activation function to produce an output. The weighted sum is calculated as follows:

$$\boldsymbol{z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b}$$

where $\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_n}$ are the weights of the connections between the inputs and the node, b is the bias term, and $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}$ are the elements of the input vector x.

The output of the node is calculated by applying an activation function f to the weighted sum:

$$\boldsymbol{a = f(z) \ \in R}$$

The output layer of the ANN model predicts the optimal move to make from the current board position. The output layer can be a softmax layer, which outputs a probability distribution over all possible moves. The probability of making move i is calculated as follows:

$$\boldsymbol{P(i) = \frac{e^{z_i}}{\sum(e^{z_j})} \ for \ all \ j}$$

where $\boldsymbol{z_i}$ is the weighted sum of the inputs to the i-th output node.

During training, the weights and biases of the ANN model are adjusted using an optimization algorithm such as gradient descent. The objective is to minimize the difference between the predicted move and the actual optimal move for a given board position. The loss function used for training might be the cross-entropy loss:

$$\boldsymbol{L = -\sum(y \times \log(\dot{y}))}$$

where y is a one-hot vector representing the actual optimal move and $\dot{y}$ is the predicted probability distribution over moves.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 13, MARCH 2023

Reinforcement Learning Algorithm

The key concept of the reinforcement learning algorithm is the interaction between the agent and the environment. In our case, the agent only knows about the possible states and actions in an environment but nothing about the state transition and reward probability functions. Therefore, we will implement a model-free RL approach (Q-learning) in a deterministic environment since we have our previously built ANN model. In a model-free RL algorithm, the agent uses experience to learn the policy or value function directly without using a model of the environment, while a deterministic environment refers to the case where both the state transition model and reward model are deterministic functions. In this process, the Q-values are stored and updated in a Q-table, which has dimensions matching the number of actions and states in the environment. Fig.5 demonstrates the design of the reinforcement learning process.

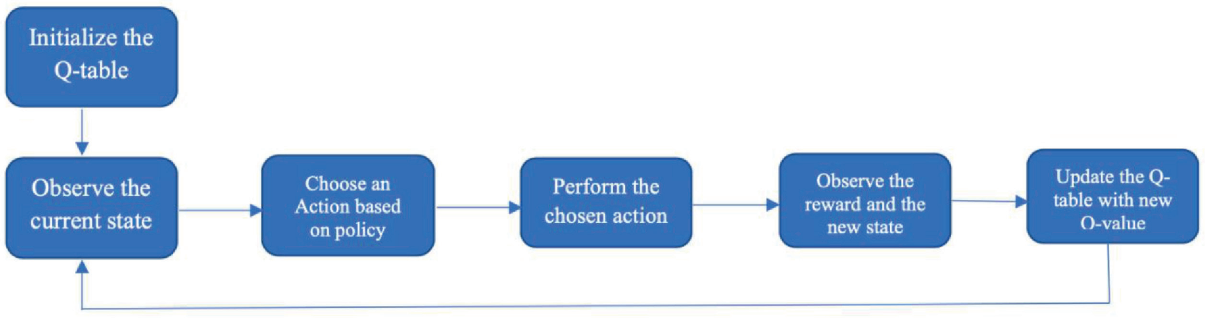
Initialize the Q-table
Observe the current state
Choose an Action based on policy
Perform the chosen action
Observe the reward and the new state
Update the Q-table with new Q-value

Figure 3. The design of the reinforcement Q-learning process

A model-free RL approach called Q-learning aims to develop a policy that instructs an agent what to do in a given state. For each specific state-action combination in this model, a Q-value exists, and these Q-values are kept in a Q-table. A higher Q-value indicates a better likelihood of receiving a larger reward.

The Q-value of getting an action a in a state s for a policy π is denoted as $Q^{\pi}(s,a)$. In an iterative process, the agent performs various actions and modifies the Q-values in the Q-table after initializing the Q-values to any value. Its Q-values are updated at each time step t in accordance with the following Bellman equation by performing the action a in the state s:

$$Q_{t+1}(s,a) = (1 - \alpha)Q_t(s,a) + \alpha[R(s,a) + \lambda\, max_{a' \in A} Q_t(s', a')]$$

here α is the learning rate $(0 < \alpha \leq 1)$, λ is the discount factor $(0 \leq \lambda \leq 1)$ that depicts the significance of future reward, and R is the immediate reward of this transition.

The exploration/exploitation trade-off requires the use of ϵ-greedy model during training and action selection. We will select the random action with ϵ probability (exploration), instead of always choosing the best learned Q-value action (exploitation). Consequently, the following model is utilized for selecting action a at state s.

$$a = \begin{cases} random\ action \in A, & \epsilon - probability \\ argmax_a(Q(s)) & otherwise \end{cases}$$

The learned values can be applied as an action-value function for the agent once the Q-values have finally converged.

Training process

The Residual neural network (ResNet) is used to train the Gobang model. It mainly consists of 3 parts: common layers, policy head and value head. Common layers are shared by the policy head and value head; the part of the policy head will compute the probability distribution of the action, and the part of value head generates the probability of winning. Among them, common layers consist of convolution blocks and residual blocks, and the policy head or value head mainly consist of convolutional layer and fully-connected layer. Considering the limit of hardware resources, we only use 2 residual blocks. The training parameters and training process are shown in Table 1 and Fig.6.

Table 1. Training parameters

Model: "sequential"		
layer (type)	Output shape	Parameter #
dense (Dense)	(None, 32)	11584
dense_1 (Dense)	(None, 64)	2112
dense_2 (Dense)	(None, 128)	8320
dense_3 (Dense)	(None, 64)	8256
dense_4 (Dense)	(None, 8)	520
dense_5 (Dense)	(None, 1)	9
Total parameters: 30,801		
Trainable parameters: 30,801		
Non-trainable parameters: 0		

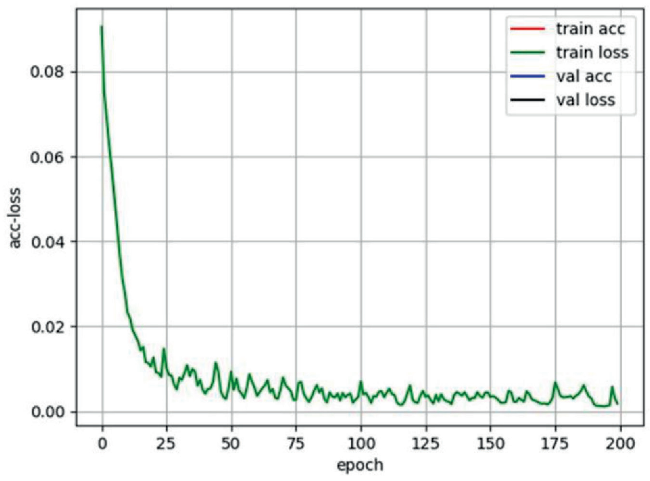

Figure 4. Model training process

The reinforcement learning algorithm used in this project is Q-learning. We enhance Q-learning with Monte Carlo Search to give QM-learning. This enhancement improves the performance of pure Q-learning. The learning procedure is demonstrated in Fig.7.

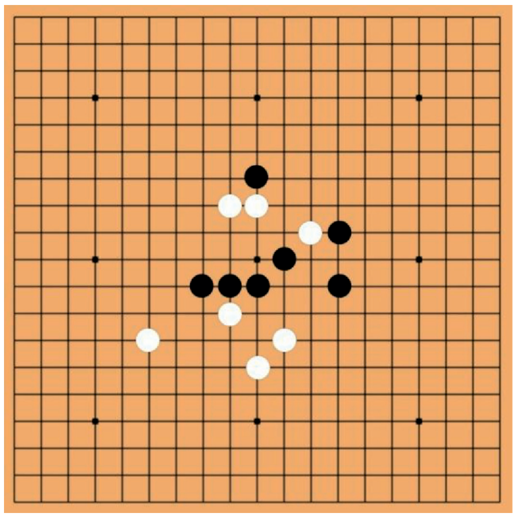

Figure 5. Performance improvement process

Conclusion

The primary objective of this paper is to implement the Artificial Intelligence based Gobang-playing algorithm with improved performance, which is faster and more efficient in its decision-making process. The article first provides an outline of the importance of the topic, followed by insights into related work from the current research, and then the algorithm is described in detail. The experiment was conducted to evaluate the performance of the algorithm, beginning with plotting the train of loss during the training process, observing the learned skill, and examining the performance while playing against a human Gobang player. The experiment results demonstrate that the trained Gobang playing algorithm analyzes the game state more quickly, considers more possible moves, and makes more accurate predictions about the likely outcome of each move, which leads to faster gameplay and better use of computational resources. Future work will be devoted to improving the modeling part so that the algorithm could be enhanced for a complex, continuous state space or action space, since the problems that are currently addressed are discrete states, which are relatively easy by nature.

References

1. Li, M., & Fu, M. (2021). Research on the Recommendation System of Gobang Game Based on Machine Learning. *The 2nd International Conference on Computing and Data Science*. https://doi.org/10.1145/3448734.3450788
2. Xie, Y., Gao, W., Dai, Z., & Li, Y. (2022). Research and Improvement of Alpha-Beta Search Algorithm in Gobang. *Advances in Transdisciplinary Engineering*. https://doi.org/10.3233/atde220084
3. Li, W. (2018). Prediction Distortion in Monte Carlo Tree Search and an Improved Algorithm. *Journal of Intelligent Learning Systems and Applications*, *10*(02), 46–79. https://doi.org/10.4236/jilsa.2018.102004
4. Jofanda, A.N.W., & Yasin, M. (2021). Design of Checkers Game Using Alpha-Beta Pruning Algorithm. *INTENSIF: Jurnal Ilmiah Penelitian Dan Penerapan Teknologi Sistem Informasi, 5(2), 279–295.* https://doi.org/10.29407/intensif.v5i2.15863
5. Sun, L., Biao, G., & Shi, H. (2022). A deep reinforcement learning method based on attentional memories. *2022 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI)*. https://doi.org/10.1109/icceai55464.2022.00108

6. Hu, J., Zhao, F., Meng, J., & Wu, S. (2020). Application of deep reinforcement learning in the board game. *Proceedings of 2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence*. https://doi.org/10.1109/ICIBA50161.2020.9277188
7. Soemers, D., Samothrakis, S., Piette, É., & Stephenson, M. (2023). Extracting tactics learned from self-play in general games. *Information Sciences, Volume 624, 277-298*. https://doi.org/10.1016/j.ins.2022.12.080
8. Li, X., Zhang, W., Chen, J., Wu, L., & Cairangdanghzhou. (2022). Gobang Game Algorithm Based on Reinforcement Learning. *Communications in Computer and Information Science, 463–475*. https://doi.org/10.1007/978-981-16-9247-5_36
9. Venkateswara Reddy, L., Saravana Kumar, S., Sugumaran, S., & Lavanya, K. (2021). Design and development of artificial intelligence (AI) based board game (Gobang) using android. *Materials Today: Proceedings*. https://doi.org/10.1016/j.matpr.2020.12.1144
10. An, Y., Primanita, A., Khalid, M., & Iida, H. (2020). Ascertaining the Play Outcome using the Single Conspiracy Number in GoBang. *2020 IEEE Conference on Games (CoG)*. https://doi.org/10.1109/cog47356.2020.9231621
11. Hu, J., Zhao, F., Meng, J., & Wu, S. (2020). Application of Deep Reinforcement Learning in the Board Game. *2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA)*. https://doi.org/10.1109/iciba50161.2020.9277188
12. Dingying, T., Huiling, L., Hanjie, L., & Pingping, C. (2022). Design and Implementation of Gobang Game Server Based on Distributed Cluster Technology. *ACM International Conference Proceeding Series, 217-224*. https://doi.org/10.1145/3517077.3517113
13. Ziang, W., Haowei, W., & Yanlong, Z. (2021). A Replay Method for Gobang Chessboard Based on Machine Vision. *Asia-Pacific Conference on Communications*. https://doi.org/10.1109/acctcs52002.2021.00060
14. C. Nunes, M. De Craene, H. Langet, O. Camara & A. Jonsson. A Monte Carlo Tree Search Approach to Learning Decision Trees. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 2018, pp. 429-435*. https://doi.org/ 10.1109/ICMLA.2018.00070