electronics

MDPI

Article

Generative Steganography Based on the Construction of Chinese Chess Record

Yi Cao [1,2,3], Youwei Du [4], Wentao Ge [4], Yanshu Huang [1,2,3], Chengsheng Yuan [4] and Quan Wang [1,2,3,*]

[1] School of Internet of Things Engineering, Wuxi University, Wuxi 214105, China; caoyi@cwxu.edu.cn (Y.C.); 20212385052@stu.cwxu.edu.com (Y.H.)
[2] Jiangsu Engineering Research Center of Hyperconvergence Application and Security of IoT Devices, Wuxi University, Wuxi 214105, China
[3] Wuxi City Internet of Vehicles Key Laboratory, Wuxi University, Wuxi 214105, China
[4] School of Computer Science, Nanjing University of Information Science & Technology, Nanjing 210044, China; 202412492570@nuist.edu.cn (Y.D.); 202312490336@nuist.edu.cn (W.G.); yuancs@nuist.edu.cn (C.Y.)
* Correspondence: wangquan@cwxu.edu.cn

Abstract: Steganography is a technique for hiding secret information in imperceptible carriers and transmitting it. Unlike traditional embedding-based steganography, generative steganography can generate stego-carriers directly from secret messages, thus avoiding modifications to natural carriers that steganalysis can detect. As a branch of generative steganography, game-behavior-based steganography transmits secret information by encoding game behavior. It can naturally integrate with real interaction scenarios, exhibiting strong concealment and undetectability. To this end, this paper proposes a generative steganography based on Chinese Chess record construction. Firstly, an AlphaZero model was trained to achieve a high level in Chinese Chess, then transmit secret information by encoding chess behavior. Specifically, in each chess step, the model generates all the current feasible moves and encodes the moves that meet the threshold strategy according to probability. Then, the appropriate move will be selected according to the secret information. To ensure the reasonableness of the generated chess records, this paper controlled the game process and designed a database of fixed opening chess records. The proposed method can hide an average of 413 bits of information for each carrier and effectively resist common image attacks. Regarding anti-steganalysis, the proposed method achieved accuracy rates of 0.498 and 0.497 on XuNet and YeNet, respectively, outperforming other behavior-based steganography techniques.

Keywords: information security; information hiding; steganography; Chinese Chess; reinforcement learning

Academic Editor: Zbigniew Kotulski

Received: 20 November 2024
Revised: 18 January 2025
Accepted: 22 January 2025
Published: 23 January 2025

Citation: Cao, Y.; Du, Y.; Ge, W.; Huang, Y.; Yuan, C.; Wang, Q. Generative Steganography Based on the Construction of Chinese Chess Record. *Electronics* **2025**, *14*, 451. https://doi.org/10.3390/electronics14030451

1. Introduction

Steganography is a technique that hides secret information into imperceptible carriers such as images [1], text [2], audio [3], and videos [4]. Unlike traditional cryptography, steganography conceals the information and the fact that the information has been concealed, ensuring the security of secret information during communication.

According to different implementation methods, steganography can generally be categorized into three common types: cover-modified [5–8], cover-selected [9,10], and cover-generated [1,10–20]. Cover-modification methods modify existing carriers to hide information. As a traditional technique in cover-modified steganography, the Least Significant Bit (LSB) [5] method embeds secret information by altering the least significant bit of

each pixel in an image. However, this method can change the statistical characteristics of the cover image, making it detectable by steganalysis [21]. Consequently, some researchers design distortion functions to modify image pixels [6,7] to minimize the impact on the image's statistical properties. With the development of deep learning, some researchers train neural networks to generate these distortion functions automatically [8]. Cover-selection methods map secret information onto specific carriers through specific algorithms, with these carriers representing the secret information. Zhang et al. [9] employed the Discrete Cosine Transform (DCT) and Latent Dirichlet Allocation (LDA) to classify images in an image database. Subsequently, the secret information could be transformed into corresponding feature sequences to identify the appropriate images for steganography. While cover-selection methods do not alter the cover images, constructing an extensive image database that meets typical communication requirements and conforms to everyday communication habits is undoubtedly challenging. Cover-generation methods use secret information to create stego-carriers directly. With the advancement of deep learning, the emergence of many generative models has facilitated the development of generative steganography. Hu et al. [1], Yang et al. [11], and Peng et al. [12] mapped secret information to a latent space and then used a Generative Adversarial Network (GAN) to generate stego-images directly. Building on this, Wen et al. [14] designed a Reversible Data Hiding (RDH) technique, transforming secret images into any chosen reference image to enhance transmission security. Zhou et al. [15] utilized the reversibility of the Glow [22] model to improve the extraction accuracy and the quality of the generated secret images. Li et al. [18] proposed a novel steganography pipeline called GaussianStego, which innovatively hides secret information into generated 3D Gaussian renderings.

With the development of social networks, behavior-based steganography has attracted considerable research interest. Zhang et al. [23] used pseudo-randomly generated binary vectors to perform inner products with secret information. By controlling "liking" behaviors on social networks, they transmit secret messages at a specific rate. Gao et al. [24] further introduced a 0–1 knapsack algorithm for personnel allocation protocols, controlling the recipient's status to enhance the efficiency and flexibility of covert transmission. Additionally, Zhou et al. [25] designed a sequence generation method based on a transition probability graph (TPG), enabling the sender to select a set of carrier data highly relevant to the secret information in product recommendation applications. Gaming, as a common form of entertainment, has also been scrutinized by several researchers. Around the 2nd century B.C., a Greek named Polybius devised a method of encoding letters into pairs of symbols for confidential communication. After that, steganography based on gaming behaviors has been continuously proposed. Mahato et al. [26] propose encoding secret information into the locations of "mines" in the "Minesweeper" game for information transmission, allowing both communicating parties to interact as gaming enthusiasts. Similarly, Ou et al. [27] encoded secret information into the sequence of different shapes of blocks generated in "Tetris" for steganography. In contrast to image steganography, this behavior-based steganography is less susceptible to conventional multimedia attacks. Still, these methods are either not generic in terms of rules, which need to be constantly updated by both communicating parties to ensure security or the behaviors are simpler and do not have a lot of room for variation. Hence, they tend to be lower in hiding capacity and have less practical applicability.

With the rapid development of machine learning and reinforcement learning technologies, computers have been able to learn very complex board games. For example, advanced reinforcement learning models such as AlphaGo [28] and AlphaZero [29] have enabled computers to achieve breakthroughs in Go and Chess. Among them, AlphaZero is a revolutionary artificial intelligence program developed by DeepMind. Launched in 2017,

it quickly achieved the ability to surpass the top human level in games such as Go, Chess, and Shogi. Its core is a deep neural network that learns the game rules from scratch and gradually improves its strategy through self-playing reinforcement learning.

For both common cover-modification and cover-generation steganography methods, the stego-carriers are either modified or generated by models to hide information, making them susceptible to detection by advanced steganalysis and inherently less secure. Meanwhile, common cover-selection steganography methods and steganography based on simple game behaviors often suffer from low hiding capacity.

To address these issues, the proposed method leveraged AlphaZero and the rules of Chinese Chess to propose a generative steganography method by constructing Chinese Chess records. The main contributions of this paper are as follows:

(1) By generating a large amount of self-play data, an AlphaZero model has been trained to achieve a high level of proficiency in Chinese Chess;
(2) Based on the trained AlphaZero model, the proposed method developed a steganography method that enables two communicating parties to achieve covert communication by transmitting chess records generated by the model;
(3) The experimental results indicate that the proposed method surpasses several existing methods in hiding capacity. It also exhibits outstanding robustness and security, and has practical feasibility and value.

The organization of this paper is as follows. The proposed steganography method is described in detail in Section 2. The experiments and analysis are presented in Section 3. The conclusion is given in Section 4.

2. The Proposed Method

Chinese Chess is a two-player board game that originated in China. The board consists of nine parallel horizontal lines and ten parallel vertical lines, totaling 90 intersections. Each side has 16 pieces: a marshal, two advisors, two bishops, two knights, two rooks, two cannons, and five soldiers. The game's object is to put the opponent's marshal to death. Both players follow specific rules and paths by moving pieces around to capture the opponent's pieces and kill the opponent's marshal.

The generative steganography method based on Chinese Chess records proposed in this paper mainly consists of four parts: (1) Training AlphaZero; (2) Constructing a database of common fixed opening chess records; (3) Generating steganographic chess records; (4) Extracting information from the steganographic chess records. The overall process is shown in Figure 1.

First, AlphaZero significantly enhances its chess-playing capabilities through extensive self-play. Once a highly proficient AlphaZero model is developed, it generates a database of common fixed opening chess records. This database serves two purposes: (1) to improve the reasonableness of the chess records and (2) to record the total length of the secret information.

During the secret information hiding phase, an opening chess record is selected based on the length of the binary secret information. Subsequently, AlphaZero initiates self-play from this opening. In each move, all feasible moves and their probabilities are generated. A threshold strategy is employed to eliminate less optimal moves, resulting in a candidate set. Next, the binary secret information is segmented and converted into decimal values. The move corresponding to the decimal index in the candidate set is executed, thereby concealing the binary information. This process is repeated until all the secret information is concealed.

During the information extraction phase, the receiver retrieves the chess record, extracts the opening chess record to determine the length of the secret information, and

performs the reverse hiding process. Each move is decoded back into fragments of the secret information. Finally, the extracted fragments are concatenated to reconstruct the complete secret information.

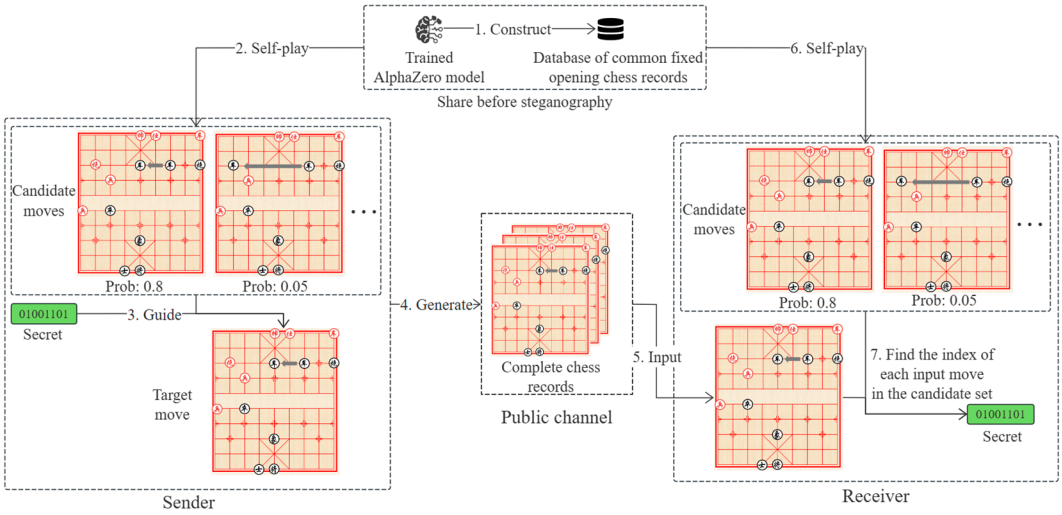

Figure 1. The overall process of the proposed steganography method based on Chinese Chess.

2.1. Training AlphaZero

In the steganographic chess records generation process, all moves are made by a fully trained AlphaZero engaged in self-play. The model can analyze the current state, evaluate the quality of all legal moves within the state, and determine the probability of selecting each move based on its quality. The principle is as follows:

AlphaZero constructs a neural network with parameters denoted as θ, represented as:

$$(policy,\ value) = f_\theta(s,\ m) \tag{1}$$

The input consists of any chess state s and the set of legal next moves m. The output includes the probability distribution $policy$ for each move within the set m, as well as the win rate $value$ for one player (usually the first player) in the current state. The $value$ is a scalar ranging from -1 to 1. The $value$ closer to -1 indicates a lower win rate for that player, whereas the $value$ nearer to 1 signifies a higher win rate.

In addition to neural networks, the Monte Carlo Tree Search (MCTS) [30] algorithm is also an essential component of AlphaZero. First, a game tree is constructed, where each node represents a game state. The root node represents the starting state of a game, and the leaf nodes represent the end states. All the child nodes of one node represent all feasible next states under the current state, and the edges connecting one node and its child nodes represent the actions of selecting a particular child state. For any node s and its child node a, the edge connecting the two nodes records three values: the number of visits $N(s,a)$, the average value $Q(s,a)$, and the selection probability $P(s,a)$ of a. The MCTS process can be divided into four steps, as shown in Figure 2.

(1) Selection. Select the optimal child node from the root node and recursively continue this process until a leaf node is reached. MCTS uses the Upper Confidence Bound (UCB) formula to determine which child node to select specifically:

$$\hat{a} = \text{argmax}_a\{Q(s,a) + cP(s,a)\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}\} \tag{2}$$

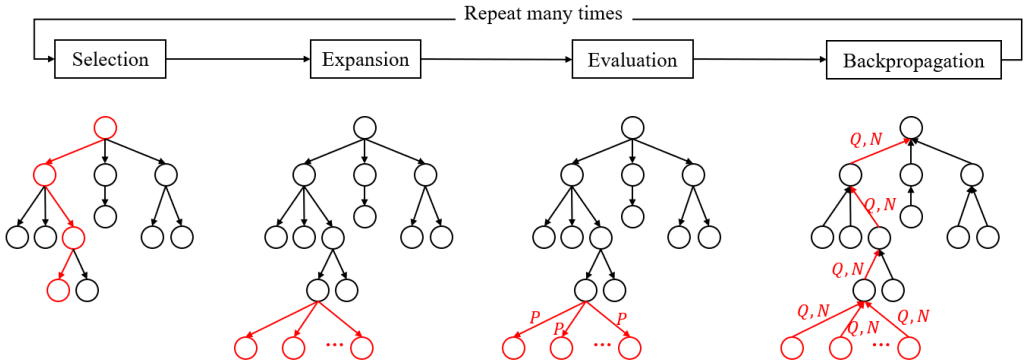

Figure 2. The Monte Carlo Tree Search Process Used by AlphaZero. The nodes and edges highlighted in red in the figure indicate that they are being manipulated.

This formula takes into account both the value of each child node itself and the child nodes that have been visited less frequently. Here, c is a constant used to balance these two factors. b denotes all the child nodes of state s. $\hat{a}$ is the child node which is finally selected.

In Formula (2), $Q(s,\ a)$ represents the value evaluation of child node a of s, aiming to consider the child node that has achieved the highest win rate based on all previous simulations. This process is referred to as "exploitation". However, other child nodes of s may possess greater potential advantages than node a, particularly if they have been selected infrequently, resulting in an undervaluation of their true worth. The UCB formula addresses this issue by assigning higher weights to less frequently selected child nodes. This aspect is termed "exploration". The constant c serves to balance these two components effectively. In the proposed method, c is set to 5.

(2) Expansion. Compute all possible next moves for the chess state represented by the leaf node, and expand them as child nodes under the current node. Additionally, initializing $N(s,a)$, $Q(s,a)$, and $P(s,a)$ to zero for each edge.

(3) Evaluation. Utilize the neural network to predict the *policy* and win rate *value* of each child node that is selected. Subsequently, store the *policy* to the corresponding edge's $P(s,a)$.

(4) Backpropagation. For all edges traversed along the search path, employ Formulas (3) and (4) to update their average value $Q(s,a)$ and visit count $N(s,a)$ in a bottom-up manner.

$$Q(s,a) = \frac{N(s,a) \times Q(s,a) + value}{N(s,a) + 1} \tag{3}$$

$$N(s,a) = N(s,a) + 1 \tag{4}$$

The more times the above process is repeated, the more complex the constructed game tree becomes, allowing it to handle more complex situations.

For the MCTS process, the complexity is influenced by both the depth and width of the simulations, let the depth of each simulation be d. Each simulation starts from the root node, where each selected chess state has w sub-states The optimal state is selected from these w sub-states, and the process is repeated until the leaf node is reached. Consequently, the time complexity of a single simulation is $O\left(w^d\right)$. If the number of simulations is n, the time complexity of MCTS is $O\left(n \cdot w^d\right)$.

At the beginning of the neural network training, the AlphaZero model exhibits limited strategic capabilities and is only able to make moves randomly within the rules through MCTS. By providing a substantial amount of self-play data generated by the model for learning purposes, the neural network's accuracy of predictions will gradually improve. This improvement is then utilized in MCTS, allowing the model to make more reasonable

moves. In turn, this facilitates the collection of higher-quality self-play data for further training of the neural network. By continuous iteration of this process, there is a sustained increase in the accuracy of predictions made by the neural network.

For the training of neural networks, let the time complexity of each forward propagation and backpropagation be $O(n)$. If the number of samples in each training iteration is k, the time complexity of one training is $O(nk)$.

The structure of the improved AlphaZero model this paper used is shown in Figure 3. The parameters of each model layer and the input and output dimensions are shown in Table 1. The chessboard's dimension is 10×9, and for each intersection where a piece can be dropped, this paper uses a 9-bit binary number to represent its state, so the input dimension is $9 \times 10 \times 9$.

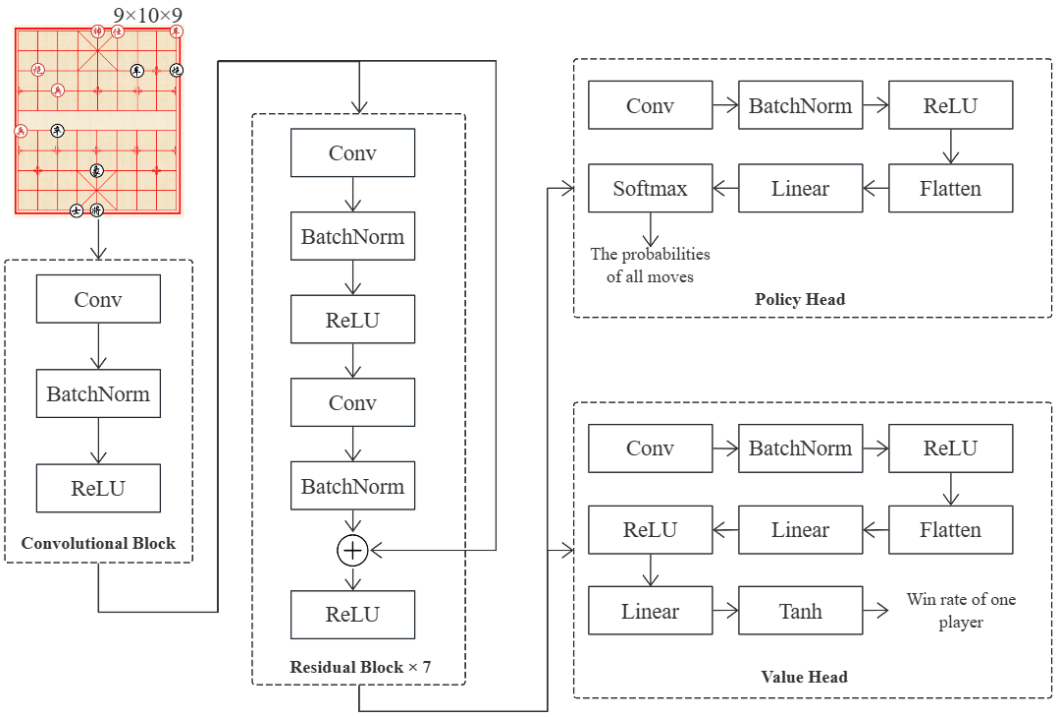

Figure 3. The structure of the improved AlphaZero model.

Table 1. The parameters of each layer of the improved AlphaZero model.

Block	Input Size	Layer	Output Size
Convolutional Block	$9 \times 10 \times 9$	Conv(3×3, 1, 1) + BN + ReLU	$256 \times 10 \times 9$
Residual Block	$256 \times 10 \times 9$ $256 \times 10 \times 9$ $256 \times 10 \times 9$	Conv(3×3, 1, 1) + BN + ReLU Conv(3×3, 1, 1) + BN Residual + ReLU	$256 \times 10 \times 9$ $256 \times 10 \times 9$ $256 \times 10 \times 9$
Policy Head	$256 \times 10 \times 9$ $16 \times 10 \times 9$ 1440	Conv(1×1, 1, 0) + BN + ReLU Flatten Linear + Softmax	$16 \times 10 \times 9$ 1440 2086
Value Head	$256 \times 10 \times 9$ $8 \times 10 \times 9$ 720 256	Conv(1×1, 1, 0) + BN + ReLU Flatten Linear + ReLU Linear + Tanh	$8 \times 10 \times 9$ 720 256 1

The backbone network of the AlphaZero model consists of a convolutional block and 7 residual blocks. The rationale for selecting this specific number of residual blocks is

detailed in Section 3.3.1. The convolutional block plays a crucial role in feature extraction from the chessboard, transforming the raw board state into a more abstract representation that enables the model to identify significant local patterns (such as threats, control areas, etc.). The residual blocks serve as core components of the model, consisting of multiple Convolution layers, BatchNorm layers, and activation functions (e.g., ReLU). The defining feature of the residual blocks is the incorporation of "residual connections", whereby the output of each block is computed as the sum of its input and the result of the convolution operation. This structure allows faster information flow through the network, alleviating the vanishing gradient problem in deep networks, thereby enhancing the model's ability to learn complex strategies and value functions.

As the neural network's output, the policy head consists of a fully connected layer that processes the outputs generated by the backbone network to produce a probability distribution to each legal move on the board. Similarly, the value head also comprises a fully connected layer that takes feature representations derived from the backbone network and outputs a scalar value representing an estimated win probability for the current board state, i.e., predicting a player's likelihood of winning from the current position.

In Table 1, "Conv" refers to the convolutional layer. The first parameter specifies the size of the convolutional kernel, the second parameter indicates the length of the stride of the kernel's movement, and the third parameter denotes the width of pixel padding applied at the edges of the feature map. The primary function of the convolutional layer is to extract and learn the features of the chessboard. "BN" represents BatchNorm layers, which normalize input data for each layer to ensure that it maintains a consistent distribution across all layers. This normalization process accelerates network convergence. ReLU is a widely utilized activation function that converts any input less than or equal to zero into zero. This non-linear transformation introduces non-linearity into the network during training, thereby enhancing its generalization capabilities.

2.2. Constructing a Database of Common Fixed Opening Chess Records

In the steganography process, it is necessary to dynamically select which move to make based on the secret information. This makes it challenging for each chosen move to be within a reasonable range. If a highly unreasonable move is made, an attacker can easily detect it, leading to communication failure. Therefore, this paper has constructed a database that saves many reasonable opening chess records to ensure reasonableness at the start of the game. The method to ensure reasonableness during the game process will be introduced in Section 2.3. The index of each opening chess record in this database can also be used to represent the length of the secret information, which is crucial for the extraction process. This database should be shared between the communicating parties before the steganography begins.

This paper utilizes the fully trained AlphaZero model to conduct a large amount of self-play automatically. For each game generated, this paper only records the first k moves. Since the opening chess records do not hide data, k should be as small as possible. For example, this paper set k to 5. During the generation process, each step discards unreasonable moves and randomly selects from all reasonable moves, with probabilities for each move generated by the neural network. Finally, all generated chess records are sequentially numbered starting from 1 and stored in the database, forming the opening chess records database. All generated opening chess records should ensure no repetition to avoid conflicts.

2.3. Generating Steganographic Chess Records

The process of generating steganographic chess records is shown in Algorithm 1.

Algorithm 1: The process of generating steganographic chess records.

INPUT: The secret message; The database of common fixed opening chess records DB
OUTPUT: One or more steganographic chess records in the form of consecutive images or a single video
START
Step 1: Convert the secret information into binary, denoted as *Secret*, and calculate its length, denoted as *len*;
Step 2: Find the opening chess record in DB with an index equal to *len*, and have the AlphaZero execute that opening record;
Step 3: The AlphaZero continues self-play, and at each step of the game, it calculates the set of all feasible moves under the current state, denoted as $L = \{x_1, x_2, \ldots, x_n\}$, and provides the execution probability ϵ_i $(1 \leq i \leq n)$ for each move;
Step 4: To ensure the reasonableness of the chess records, specific measures need to be taken to remove some unreasonable moves while ensuring that the number of removed moves does not excessively reduce the hiding capacity. This paper uses the following method to remove some moves from L:

(1) Identify the move in L with the highest probability and denote its probability value as $maxprob$;
(2) If $maxprob \geq \alpha$, retain only that move and discard all others;
(3) If $\beta \leq maxprob < \alpha$, remove all moves that satisfy $\epsilon_i < \gamma$;
(4) Remove all moves that satisfy $\epsilon_i = 0$.

The triplet (α, β, γ) is referred to as the "threshold strategy", and its selection method will be detailed in Section 3.2.
Step 5: Let the number of remaining moves in L be n. If $n = 1$, it indicates that data cannot be hidden in this step, so execute this only move directly and jump to **Step 7**. Otherwise, it suggests that this step can hide $\lfloor log_2 n \rfloor$ bits of binary data. Sort the remaining moves in descending order by probability and number them starting from 0.
Step 6: Take a segment from the beginning of *Secret* with a length of $\lfloor log_2 n \rfloor$, convert it into a decimal number *dec*, find the move in L with the index equal to *dec*, and execute that move. Specifically, if the remaining length of *Secret* is less than $\lfloor log_2 n \rfloor$, extract all remaining content from *Secret*.
Step 7: The AlphaZero continuously repeats **Step 3 to 6** until *Secret* is entirely hidden. Screenshots of each move in the game process are taken and sequentially combined into a series of continuous images or a video, resulting in the final steganographic chess record.
END

The process of generating a steganographic chess record is shown in Figure 4.

It should be noted that since one function of the opening chess record is to record the secret information's length, the secret binary data *len* must not exceed the number of chess records in the database. Specifically, if the game ends before the steganography is hidden, a new game must be started to hide the remaining data. The specific logic is as follows:

If a single game cannot completely hide all the data, start a new game to conceal the remaining data. For the remaining data, fully execute Algorithm 1. The process is then repeated until all the data are hidden. It is easy to see that in the obtained multiple steganographic chess records, the index of each opening record in the database DB must be decreased because it represents the length of the remaining data. During the steganography process, the length of the secret information must be decreased. When receiving multiple steganographic chess records, the recipient can use this fact to sort them in descending

order by the index of the opening chess records to determine the order. This method somewhat reduces interference from network factors or attackers, improving the robustness of steganography.

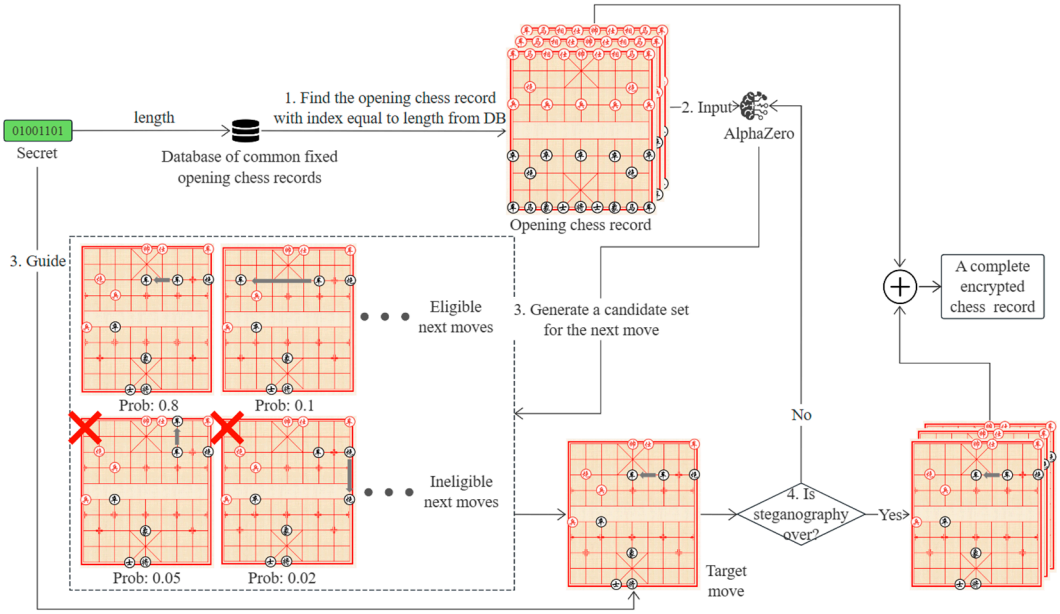

Figure 4. The process of generating a steganographic chess record. The chess records marked with a red "×" in the figure are those that have been deleted for failing to meet the threshold strategy.

For the process of generating chess records, each move needs to go through the same MCTS as the training process, its complexity is $O\left(n \cdot w^d\right)$, and then all the current feasible moves and their probabilities can be given. For a chess game, let the total number of moves be m. Then the time complexity of generating a complete game is $O\left(m \cdot n \cdot w^d\right)$.

2.4. Extracting Information from the Steganographic Chess Records

The extraction process of the steganographic chess records is shown in Algorithm 2.

The extraction process of one steganographic chess record is shown in Figure 5. It should be noted that when extracting information from the last chess record's last move, there may be a discrepancy between the length of *result* and the actual length of the original secret information. For example, if the original secret information corresponding to the last move is "011" and this move can hide 5 bits of data, then after extraction, it would become "00011" to satisfy the required length of 5 bits. This representation is incorrect. At this point, the length of the secret data *len*, which is previously recorded, comes into play. For the final segment of binary information extracted, a certain number of leading zeros should be removed until the total length of *result* equals *len*.

In the example above, suppose the total length of the secret information is 200 bits. By this penultimate step, the extracted length of the secret information is 197 bits, and the hiding capacity of the last step is 5 bits, with the extracted information being "00011". Consequently, combining these 197 bits already obtained with the 5 bits from this final step results in a cumulative total of 202 bits—this does not align with the known total length for secret information (200 bits). Therefore, the first 2 leading zeros of the last segment of binary data should be removed, transforming it into "011" so that the final extracted secret information length matches the 200 bits.

Algorithm 2: The extraction process of the steganographic chess records.

INPUT: One or more steganographic chess records; The database of common fixed opening chess records DB; The extracting result cache variable $result$
OUTPUT: The secret message
START
Step 1: If the number of received steganographic chess records is greater than 1, examine the indices of the first k moves of each record in DB, sort all steganographic chess records in descending order of the opening chess record indices, and decrypt them in order. At the same time, record the index of the opening chess record with the largest index, which is the length of the complete secret information, denoted as len;
Step 2: Let N be the current steganographic chess record being decrypted. The AlphaZero executes its first k moves and removes them from N;
Step 3: The AlphaZero continues self-play, and at each step of the game, it calculates the set of all feasible moves under the current state, denoted as $L = \{x_1, x_2, \ldots, x_n\}$, and provides the execution probability ϵ_i $(1 \leq i \leq n)$ for each move;
Step 4: To ensure the reasonableness of the chess records, specific measures need to be taken to remove some unreasonable moves while ensuring that the number of removed moves does not excessively reduce the hiding capacity. This paper uses the following method to remove some moves from L:

(1) Identify the move in L with the highest probability and denote its probability value as $maxprob$;
(2) If $maxprob \geq \alpha$, retain only that move and discard all others;
(3) If $\beta \leq maxprob < \alpha$, remove all moves that satisfy $\epsilon_i < \gamma$;
(4) Remove all moves that satisfy $\epsilon_i = 0$.

Step 5: Let the number of remaining moves in L be n. If $n = 1$, it indicates that data cannot be hidden in this step, so execute this only move directly and jump to **Step 8**. Otherwise, it suggests that this step can hide $\lfloor log_2 n \rfloor$ bits of binary data. Sort the remaining moves in descending order by probability and number them starting from 0.
Step 6: Take the first move from the remaining moves of the chess record N, check its index dec in L, and convert dec into a binary number bin. If the length of bin is less than $\lfloor log_2 n \rfloor$, pad with zeros in front until its length equals $\lfloor log_2 n \rfloor$;
Step 7: Delete this move from the chess record N, and let $result += bin$;
Step 8: Repeat **Steps 3 to 7** until the current steganographic chess record is empty, indicating that decryption is complete, and proceed to decrypt the next chess record;
Step 9: Repeat **Steps 2 to 8** until all steganographic chess records have been decrypted. The $result$ is the original secret information.
END

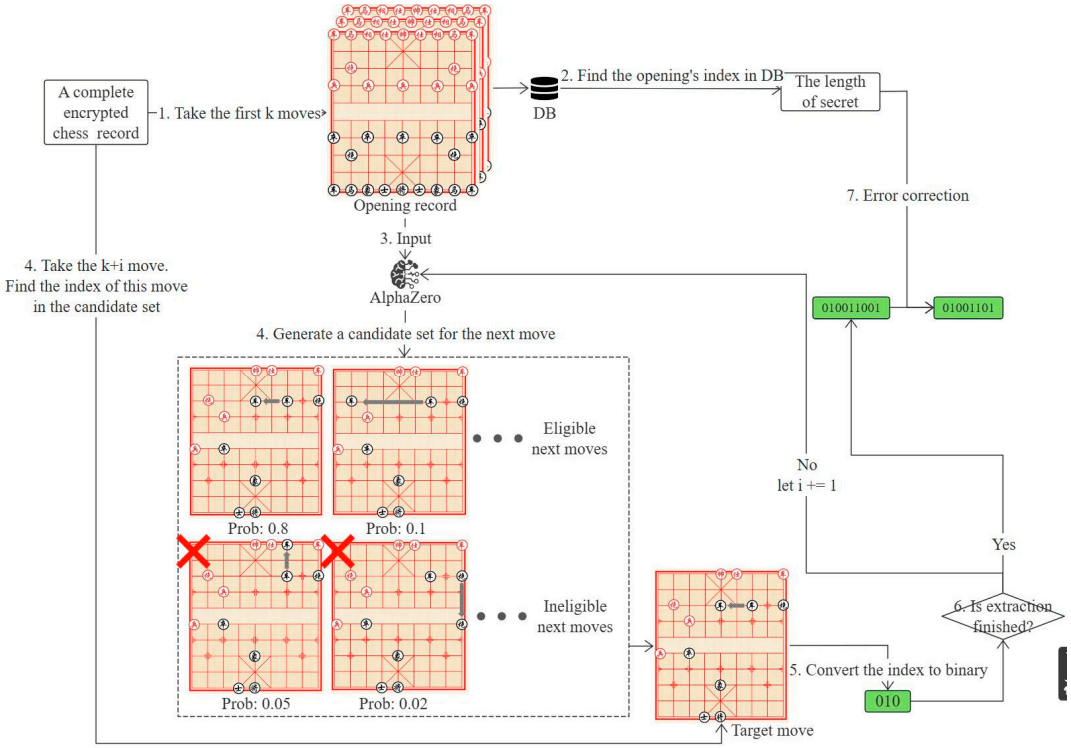

Figure 5. The extraction process of one steganographic chess record. The chess records marked with a red "×" in the figure are those that have been deleted for failing to meet the threshold strategy.

3. Experimental Results and Analysis

3.1. Experimental Setup

The experiments in this paper use the PyTorch 1.10.2 framework, equipped with an Intel Xeon E5-2680 v4 CPU, 32 GB of RAM, and an NVIDIA GeForce RTX 3080 Ti GPU. This hardware platform is rented from a Chinese computing resource provider named AutoDL. The optimizer chosen is Adam, with hyper-parameters set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay is set to 0.002. The initial learning rate is 0.001, and the learning rate is dynamically adjusted using KL divergence.

3.2. Evaluation Criteria

In the experiment, this paper uses the following criteria to evaluate the performance of the proposed method in various aspects.

3.2.1. Hiding Capacity

Since the steganographic chess records generated by the proposed method can take multiple forms, such as text records of the game, screenshots of each move in the game, or a video of the game process, this paper selected a more general metric, which is to calculate the average number of bits of data that can be hidden per carrier.

During the game process, in addition to the opening record, this paper assumes that a total of p steps were made, and q_i $(1 \leq i \leq p)$ represents the number of available moves at the i-th step. The total hiding capacity is shown in Formula (5).

$$total\_capacity = \sum_{i=1}^{p} \lfloor log_2 q_i \rfloor \tag{5}$$

3.2.2. Extraction Accuracy

Extraction accuracy measures the precision of recovering the original information from the stego-carrier, reflecting the fidelity of the steganography algorithm in preserving the original data during the hiding process. This metric is determined by comparing the data sequences before and after steganography and calculating the proportion of matching bits. Specifically, this paper uses the bit error rate (BER) to calculate the differences between the secret binary information before and after steganography. Let the original binary data be denoted as A, its total length be denoted as M, and the binary information extracted after steganography be denoted as B. A_i and B_i $(1 \leq i \leq M)$ represent the i-th binary digit of A and B, respectively. The bit error rate is then represented by Formula (6):

$$BER = \frac{\sum_{i=1}^{M} A_i \oplus B_i}{M} \times 100\% \quad (6)$$

In this formula, the "$\oplus$" symbol represents the XOR operation. For two binary numbers A and B, $A \oplus B$ denotes that for each corresponding bit position of A and B, the result is 1 if and only if the bits are different; otherwise, the result is 0. The lower the bit error rate, the higher the extraction accuracy.

3.2.3. Robustness

Robustness measures the extraction accuracy of the stego-carrier after it has been subjected to attacks, which is crucial for determining whether a system can be practically applied. For steganographic chess records in the form of continuous images or videos, this paper calculates the extraction accuracy after they have been subjected to common image attacks. Some common image attack methods are shown in Figure 6.

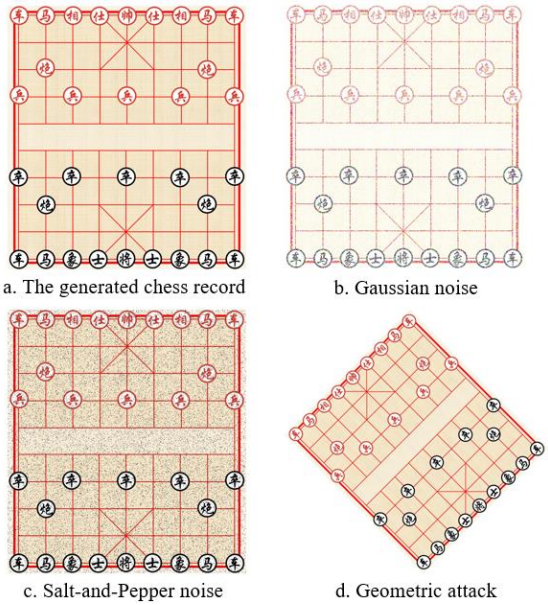

Figure 6. Some common image attack methods.

3.2.4. Security

Security measures the ability of the stego-carrier to resist analysis by professional steganalysis. For images or videos containing secret information, steganalysis can analyze their statistical characteristics at the pixel level to determine whether they contain secret information. This paper selected XuNet [31] and YeNet [32] to test their anti-steganalysis ability for steganographic chess records in continuous images or videos. To train these two types of steganalysis, this paper first performed many games without hiding information,

following the normal game process, and intercepted 5000 different game images from them as the real dataset. Secondly, this paper played several games with the purpose of hiding information and intercepted 5000 different images of chess games as the stego-image dataset. Next, this paper used these two datasets as adversarial samples to train XuNet and YeNet.

For the steganalysis, this paper uses the false alarm ratio P_{FA} to denote its probability of detecting incorrectly for the carriers without secret information. Let the total number of carriers without secret information be NC, and the number of them that are correctly detected be NCT, then P_{FA} is:

$$P_{FA} = \frac{NC - NCT}{NC} \tag{7}$$

This paper uses the missed detection ratio P_{MD} to denote its probability of incorrect detection for carriers containing secret information, and let the total number of carriers containing secret information be NS, where the number of correctly detected is NST, then P_{MD} is:

$$P_{MD} = \frac{NS - NST}{NS} \tag{8}$$

This paper uses the minimum average decision error ratio P_E to represent the overall performance of the steganalysis:

$$P_E = \frac{P_{FA} + P_{MD}}{2} \tag{9}$$

It is evident that the closer P_E is to 50%, the less detectable the steganalysis becomes, and it is almost impossible to sense the carrier's containment, which proves that the carrier has excellent resistance to steganalysis.

3.2.5. Reasonableness of the Generated Chess Record

It is essential to quantify the reasonableness of the chess records generated by AlphaZero. It is known that the neural network in the AlphaZero model can predict the final win rate resulting from a specific move, with the win rate ranging from $[-1, 1]$. This paper uses Formula (10) to quantify the reasonableness of a generated chess record.

$$R = \frac{1}{N} \sum_{i=1}^{N} \min(\Delta S_i, 0) \tag{10}$$

Here, R represents the reasonableness score of a chess record, N denotes the total number of moves in the record and ΔS_i represents the change in win rate after the i-th move $(1 \leq i \leq N)$. If the i-th move results in a decrease in the win rate, then $\Delta S_i < 0$; otherwise, $\Delta S_i \geq 0$.

In the steganography task, only highly unreasonable moves are likely to raise suspicion from an attacker. Therefore, this paper focuses only on each move's negative impacts on the chess record's overall reasonableness. Moves that improve the win rate are ignored in the formula, which is why the minimum function is used to constrain ΔS_i. Consequently, the closer the final reasonableness score R approaches 0, the more reasonable the chess record becomes; conversely, as it nears -1, its reasonableness diminishes correspondingly.

3.3. Ablation Experiment

To explain how to select the optimal parameters for the model when performing the steganography task, this paper initially sets aside the steganographic component and concentrates exclusively on the model's performance in Chinese Chess gameplay. This approach entails that each move is determined based on the highest probability at each

step during gameplay, without being limited by a threshold strategy. Within the limits of the experimental conditions, the model with the most training is chosen as the baseline. Subsequently, the model's parameters are adjusted, and comparisons are made regarding hiding capacity and the reasonableness of the generated chess records. Finally, the model that performs best in the steganography task is selected.

In the experiment, this paper sets the number of MCTS simulations per move to 1600, the epoch of training to 1200, and the number of residual blocks in the neural network to 11. The model trained with these settings is used as the baseline model. The baseline model is then used for simulated games, and its reasonableness score is evaluated using its value network. Since the evaluator and the subject are the same model, it is evident that $R_{baseline} = 0$.

Next, the number of MCTS simulations and residual blocks are gradually reduced, and the reasonableness of these models is evaluated using the neural network of the baseline model. The results are shown in Figure 7.

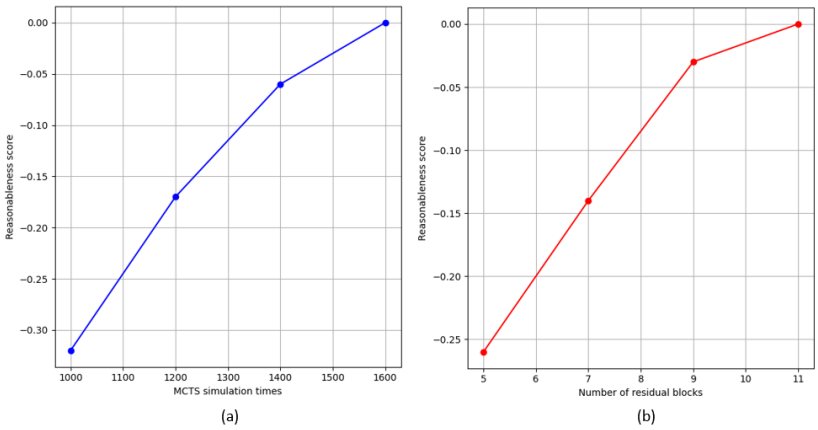

Figure 7. (**a**) The influence of different MCTS simulation times on the reasonableness of generating chess records; (**b**) The influence of the different number of residual blocks on the reasonableness of generating chess records.

It can be observed that the higher the values of the above two parameters, the better the performance of the trained model and the more reasonable the generated chess records.

However, better model performance does not necessarily mean it is more suitable for the steganography task. For example, suppose there is a near-perfect model that can provide the optimal move with almost 100% certainty for every move. In that case, it becomes tough to implement an automated threshold strategy that allows the candidate set size for each move to exceed 1. In this extreme case, the hiding capacity would be 0.

3.3.1. The Number of MCTS Simulations and Residual Blocks in AlphaZero

To find suitable model parameters that balance the reasonableness of the generated chess records and the hiding capacity, this paper fixed the number of training iterations to 1200 and set the candidate set thresholds to $(0.8,\ 0.3,\ 0.01)$. Subsequently, this paper conducted comparative experiments on different combinations of the number of MCTS simulations and residual blocks in the AlphaZero network. For each parameter combination, 50 games were played, and the evaluation metrics included the average reasonableness score of the generated chess records and the average hiding capacity. The experimental results are shown in Table 2. The corresponding heat map is shown in Figure 8.

Table 2. The influence of different MCTS simulation times and the number of residual blocks on hiding capacity and reasonableness of chess records.

MCTS Simulation Times	Number of Residual Blocks	Hiding Capacity (bits/carrier)	Reasonableness Score
1600	11	218	−0.02
1600	9	251	−0.05
1600	7	312	−0.16
1600	5	344	−0.31
1400	11	237	−0.06
1400	9	309	−0.09
1400	7	365	−0.20
1400	5	398	−0.30
1200	11	260	−0.18
1200	9	372	−0.21
1200	7	413	−0.20
1200	5	450	−0.33
1000	11	302	−0.33
1000	9	366	−0.35
1000	7	440	−0.39
1000	5	487	−0.41

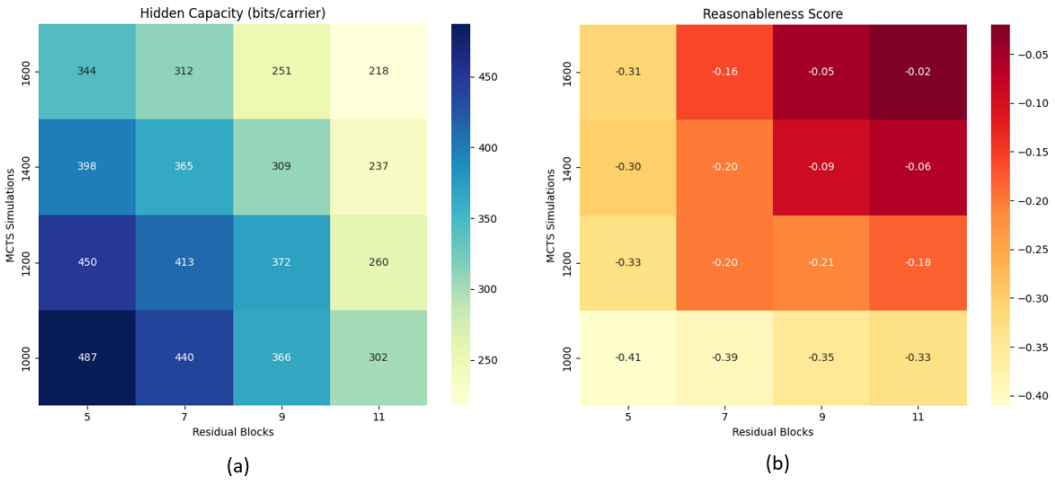

Figure 8. (**a**) The influence of the different combinations of MCTS simulation times and the number of residual blocks on the hiding capacity; (**b**) The influence of the different combinations of MCTS simulation times and the number of residual blocks on the reasonableness score of chess records.

If the reasonableness score is restricted to above −0.2, the optimal solution is a model with 1200 MCTS simulations and 7 residual blocks, achieving a hiding capacity of 413 bits/carrier. However, the hiding capacity will be significantly reduced if better reasonableness is pursued. Considering the reasonableness of the generated chess records and the hiding capacity, this paper selected the model with 1200 MCTS simulations and 7 residual blocks as the final model.

3.3.2. The Threshold Strategy

The threshold strategy for generating the candidate set mentioned in Section 2.2 is also critical. The proposed method uses a threshold strategy represented by a triplet $(\alpha, \beta, \gamma)(0 \leq \gamma \leq \beta \leq \alpha \leq 1)$, with the following steps for generating the candidate set:

(1) Identify the move in L with the highest probability and denote its probability value as $maxprob$;
(2) If $maxprob \geq \alpha$, retain only that move and discard all others;
(3) If $\beta \leq maxprob < \alpha$, remove all moves that satisfy $\epsilon_i < \gamma$;
(4) Remove all moves that satisfy $\epsilon_i = 0$.

Suppose the probability of a certain move is very high (greater than α). In that case, it indicates that the model is highly confident in this move, and its confidence level is significantly higher than for other moves (e.g., the move might directly lead to victory). Failing to make this move might be interpreted by an attacker as "deliberate underperformance", raising suspicions about potential secret information being transmitted during the game.

If no significantly good moves exist among the feasible options, but there are bad moves (with probabilities less than γ), these bad moves must be removed to avoid raising suspicions from an attacker due to their unreasonableness.

The threshold strategy choice affects the reasonableness of the chess records and the hiding capacity. If the strategy is too strict, it will result in very few candidates, significantly reducing the hiding capacity. If it is too lenient, it will lead to insufficient reasonableness of the chess records. Therefore, this paper tested different threshold strategies based on the final model and conducted comparative experiments. The experimental results are shown in Table 3.

Table 3. The influence of different threshold strategies on hiding capacity and reasonableness of chess records.

Serial Number	Threshold Strategy	Hiding Capacity (bits/carrier)	Reasonableness Score
1	(0.9, 0.3, 0.01)	450	−0.31
2	(0.8, 0.3, 0.01)	413	−0.20
3	(0.7, 0.3, 0.01)	380	−0.17
4	(0.6, 0.3, 0.01)	325	−0.15
5	(0.8, 0.2, 0.01)	397	−0.18
6	(0.8, 0.4, 0.01)	431	−0.23
7	(0.8, 0.5, 0.01)	449	−0.27
8	(0.8, 0.3, 0.005)	426	−0.22
9	(0.8, 0.3, 0.05)	398	−0.17

Based on experiments 1, 2, 3, and 4, it can be observed that as the threshold α increases, the hiding capacity will also rise. However, this leads to a decrease in the reasonableness of the chess records. This is because a larger α causes moves that have not exceeded α in probability but are advantageous not to be stably selected. Instead, they are grouped with other moves of low probability to form a candidate set, thereby reducing reasonableness. At the same time, an excessively small α, while increasing the reasonableness of the chess records, will cause significant damage to the hiding capacity.

Combining experiments 2, 5, 6, and 7, it can be seen that as the threshold β increases, the hiding capacity will rise slightly, but the chess records' reasonableness will significantly

decrease. This is because a larger β will cause moves that are sufficiently bad to be deleted with a lower probability, ultimately leading to a reduction in reasonableness.

Comparing experiments 2, 8, and 9, it can be observed that as the threshold γ increases, the hiding capacity slightly decreases, and the reasonableness of the chess records slightly rises. γ measures the probability of sufficiently bad moves. As γ increases, more moves will be deleted from the candidate set.

Finally, this paper chose the threshold strategy as $(0.8,\ 0.3,\ 0.01)$ and based the following comparative experiments on this.

3.4. Comparative Experiment

3.4.1. Hiding Capacity

Through experimental validation, it has been shown that when using the algorithm described in Section 2.3, even if the most disadvantageous moves are chosen each time, the game can still last for at least 10 moves before ending. Each move can hide 2 to 5 bits of binary data during this process, meaning that a single game can hide at least 20 bits of data. A game usually has 60~120 moves, and the average hiding capacity can reach 413 bits. In extreme cases, the hiding capacity of a single game can even exceed 650 bits. The comparison of results with some existing methods is shown in Table 4.

Table 4. Results of the comparison experiments on hiding capacity.

Method	Hiding Capacity (Bits/Carrier)
Zhang et al. [9]: DCT-LDA	15
Chen et al. [10]: Image-Select-StarGAN	33
Hu et al. [1]: DCGANs	37.5
Huang et al. [33]: SMH-SWE	128
Mahato et al. [26]: Minesweeper	214
Yang et al. [11]: PARIS	1219
Proposed method	413

Compared to steganography methods based on simpler games, the proposed method demonstrates a significant advantage in hiding capacity. For example, Mahato et al. [26] proposed a steganography method based on the Minesweeper game, hiding secret information into the distribution of mines. The hiding capacity is only 17.6 bits/carrier for the beginner level and increases to 214 bits/carrier for the expert level, but it still falls short compared to the proposed method. This is primarily because the complexity of Chinese Chess is much higher than that of Minesweeper, with more moves and larger candidate sets for each step providing greater space for hiding information. Additionally, the proposed threshold strategy acts as a "soft constraint", which can be dynamically adjusted to improve hiding capacity. In contrast, in Minesweeper, the placement of mines must fully comply with the rules. Any conflict with the surrounding numbers could easily raise suspicion from attackers, limiting the potential for increasing hiding capacity.

Compared to some non-game-based steganography methods, the proposed method offers certain advantages, such as methods proposed by Zhang et al. [9] and Hu et al. [1]. Zhang et al.'s method first utilizes Latent Dirichlet Allocation (LDA) to group images with similar content, selecting images belonging to the same topic to reduce the attacker's suspicion. Then, the selected images are processed with Discrete Cosine Transform (DCT), dividing each image into 8×8 sub-blocks, and feature sequences are generated based on the relationships between adjacent sub-blocks. During steganography, the secret information is divided into sequences, and a group of images with matching feature sequences is identified for information concealment.

Hu et al.'s method, on the other hand, involves training a generator and an extractor. In the steganography process, the secret information is mapped to a latent vector, which is then used to generate a stego-image. After receiving the stego-image, the recipient retrieves the secret information using the extractor.

In Zhang et al.'s method, the stego-carriers are the feature sequences of images. Due to the limitations of the feature extraction method, the length of the feature sequences is often low, and the image resolution and sub-block division further restrict the hiding capacity. For Hu et al.'s method, the hiding capacity depends on the dimensionality of the mapped latent vector. Although the quality of the generated images is high, each image can only hide a fixed-length latent vector, leading to a relatively low hiding capacity per carrier.

Yang et al. [11] built upon Hu et al.'s method with a focus on optimizing hiding capacity. They first proposed a capacity formula based on the latent space dimension and embedding rate. Then, they employed an inverse transform sampling method to precisely map uniformly distributed secret information into the latent space. This mapping approach utilized the capacity of the latent vector more efficiently, allowing every part of the latent space to carry secret information. As a result, their method significantly improved hiding capacity, surpassing the proposed method in this aspect.

However, hiding capacity is not the core advantage of the proposed method. Compared to Yang et al.'s approach, the proposed method offers superior performance regarding reasonableness and undetectability. First, both communication parties can naturally establish contact under the guise of being Chinese Chess enthusiasts, which is a common and natural identity in daily life. In contrast, for image-based steganography, the communicating parties must first create a reasonable communication context around the image content to ensure that the images exchanged appear logical within the context. This introduces additional complexity.

Second, the proposed method hides information during the gameplay process rather than in the textual or pixel data of the chess record. As long as the gameplay process appears sufficiently reasonable, the resulting game records are indistinguishable from normal ones. Attackers cannot identify differences between steganographic and non-steganographic chess records based on their text, pixels, or structure.

In contrast, stego-images generated by models inherently differ from natural images. No matter how close they come to realism, they still struggle to fully replicate natural characteristics, such as true-to-life colors and natural noise. Furthermore, these images carry hidden information, which steganalysis tools could potentially detect. Therefore, the proposed method offers clear advantages in these two aspects.

3.4.2. Extraction Accuracy

Unlike common steganography methods that modify image pixels, the proposed method is a game-behavior-based steganography approach, where the secret information does not rely on a specific carrier for its existence. Therefore, as long as the information receiver can understand the entire chess game process, they can fully reconstruct the secret information. The extraction accuracy can be considered 100% without being subjected to attacks.

3.4.3. Robustness

When using continuous images or videos for transmission, if they are subjected to common image attacks, as long as the information receiver can discern the game process with the naked eye, it will not affect the extraction accuracy. Therefore, the proposed method demonstrates a significant advantage in robustness. Table 5 compares the robustness between the proposed method and some existing methods.

Table 5. Results of the comparison experiments on robustness (BER).

Attack Method	Parameters	Chen et al. [10]	Hu et al. [1]	Yang et al. [11]	Peng et al. [12]	Cao et al. [13]	Proposed Method
Rotation	10°	1.02	31.79	0.58	45.56	0.09	0
	30°	1.38	42.25	0.84	47.33	0.12	0
	50°	2.69	46.11	0.82	51.02	0.17	0
Salt-and-Pepper Noise	σ(0.01)	3.20	34.88	6.32	10.82	0.97	0
	σ(0.05)	9.15	36.59	23.98	23.20	1.70	0
	σ(0.1)	14.22	38.15	28.86	38.09	3.29	0
Median Filtering	(3 × 3)	0	33.51	0.89	25.72	0.18	0
	(5 × 5)	1.87	36.14	2.61	37.11	0.35	0
	(7 × 7)	2.98	38.19	22.60	44.14	0.47	0
Mean Filtering	(3 × 3)	0.13	32.90	0.28	38.63	0.24	0
	(5 × 5)	0.37	33.24	1.61	44.96	0.19	0
	(7 × 7)	2.85	34.06	2.03	44.65	0.70	0
Gaussian Filtering	(3 × 3)	0	31.06	0.81	27.47	0.17	0
	(5 × 5)	1.55	31.18	1.15	32.03	0.31	0
	(7 × 7)	1.98	30.98	1.18	32.46	0.50	0
Gaussian Noise	σ(0.01)	0.14	36.28	8.39	20.29	0.54	0
	σ(0.05)	0.68	38.77	15.18	22.84	0.73	0
	σ(0.1)	1.26	39.81	21.12	41.37	0.96	0

3.4.4. Security

Similarly, when using continuous images or videos for transmission, since the secret information is not stored directly in the pixels of the image, it is not possible to make it converge when training the steganalysis. Therefore, the proposed method performs well when detected by steganalysis, as shown in Table 6.

Table 6. Results of the comparison experiments on security (P_E). The data presented in bold indicate the experimental results of the proposed method, which demonstrates superior performance.

Steganalysis	Jing et al. [34]: HiNet	Li et al. [35]: WGAN-GP	Peng et al. [12]: GAN-GD	Proposed Method
XuNet	0.365	0.542	0.528	**0.498**
YeNet	0.313	0.485	0.491	**0.497**

As can be seen, the performance of the GAN-based steganography proposed by Peng et al. in terms of resistance to steganalysis is similar to the proposed method. However, in reality, there is a fundamental difference between the two. Peng et al.'s method uses a generative model to hide secret information in images. Since the stego-images are generated by the model, they inevitably differ from natural images in aspects such as color and noise distribution. With the advancement of steganalysis techniques, they may still be detectable. In contrast, the proposed method does not hide the secret information in the pixels of the chess record image. Instead, it hides the information in the process of the game itself. Regardless of whether the chess record contains secret information, there is no difference in the probability distribution of the image pixels, making it undetectable even by the most sophisticated steganalysis.

3.5. Practical Application Case

This paper designed a visualization scenario to demonstrate the feasibility of the proposed method by uploading the steganographic chess record to a real social platform, as shown in Figure 9.

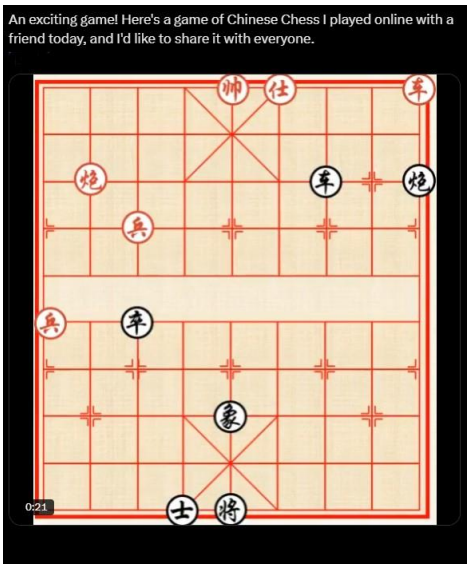

Figure 9. An example of covert communication on Twitter. The Chinese Chess game is only representation, and secret information is hidden in the process of playing chess.

The sender of the secret information disguises themselves as a Chinese Chess enthusiast and shares their game process on a public social platform. Still, the chess record they share contains hidden secret information. The receiver can establish a natural connection with the sender by subscribing and reading the posts, thereby obtaining the hidden information from the chess record.

As can be seen, the proposed method allows the stego-carrier to be seamlessly integrated into everyday life, demonstrating its feasibility and the practicality of covert communication over public channels.

3.6. Model Inference Performance Evaluation

To evaluate the computational resources the proposed method requires, this paper assessed its inference performance on various hardware platforms, with the results shown in Table 7. This paper first selected different hardware devices, including various CPUs and GPUs, as testing platforms. Then, on each platform, this paper performed 100 tasks of generating steganographic chess records and recorded the time consumed for each move. Finally, this paper calculated the average inference time based on the results.

The proposed method generally performs significantly better on GPUs compared to CPUs. Although the method does not necessarily need to operate in real-time environments, the proposed method would have certain limitations in a CPU platform if such a situation arises.

Table 7. Comparison of inference time of the proposed method on different hardware platforms. These hardware platforms are rented from a Chinese computing resource provider named AutoDL. The data in bold in the table indicate that the proposed method achieves the best performance on this platform.

GPU	Time (ms)	CPU	Time (ms)
RTX 3060 12 GB	489	Gold 6130	4135
RTX 4060Ti 8 GB	473	E5-2680 v4	3443
RTX 2080Ti 11 GB	404	AMD EPYC 7453	2650
V100 32 GB	386	Platinum 8352V	2317
A100 40 GB	360	Gold 6348	1898
RTX 3090 24 GB	295	Platinum 8457C	1418
L20 48 GB	219		
RTX 4090 24 GB	**206**		

4. Conclusions

In this paper, a fully trained AlphaZero model is utilized to automate the game of Chinese Chess by manipulating the game process to hide secret information. The proposed method mainly includes four parts: training AlphaZero, constructing a database of common fixed opening chess records, generating steganographic chess records, and extracting information from the steganographic chess records, achieving the concealment and extraction of secret information through their combination. The experimental results show that the method can generate chess records that conform to regular human behavioral habits, which can somewhat reduce the suspicion of attackers. Moreover, the method has a high hiding capacity and can adjust the hiding capacity by modifying the probability threshold of the moves, allowing for a reasonable trade-off between the reasonableness of the steganographic chess record and the hiding capacity according to different usage scenarios. In addition, this method is highly robust and secure, making it promising for practical application.

The proposed method has some limitations. Firstly, it lacks real-time performance under insufficient computational resources. While the method currently operates mainly in non-real-time environments, future work could explore how to achieve fast processing and transmission of secret information in real-time environments, thus expanding the application scenarios of the method. Furthermore, since AlphaZero is not limited to learning a specific type of chess game, we could also explore a general steganography method based on game behaviors, allowing both parties to freely choose the carrier for steganography, further enhancing the applicability and flexibility of the method.

Author Contributions: Conceptualization, Y.D. and Y.C.; formal analysis, Y.D.; investigation, Y.D., Y.C., W.G., Y.H. and C.Y.; writing—original draft preparation, Y.D., W.G., Y.H. and Y.C.; writing—review and editing, Y.D., W.G., Y.H., Y.C., C.Y. and Q.W.; visualization, Y.D.; supervision, C.Y. and Q.W. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was funded by the "'Taihu Light' Science and Technology Project of Wuxi (K20241046), Open Project of National Engineering Technology Research Center For Sensor Network (2024YJZXKFKT02, 2024YJZXKFKT01), Jiangsu Universities' General Project for Philosophy and Social Science Research (2023SJYB0919), Wuxi University Special Fund for the Introduction of Talent Research (No. 2022r043), National Natural Science Foundation of China (No. 62372125, No. 61972205, No. 62102462, No. 62102189), Postgraduate Research and Practice Innovation Program of Jiangsu Province (SJCX24_0454), and Excellent Universities' Team in Jiangsu Province for Science and Technology Innovation (Real-time Industrial Internet of Things)".

Data Availability Statement: Data is contained within the article.

Acknowledgments: The authors highly thank the National Natural Science Foundation of China.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Hu, D.; Wang, L.; Jiang, W.; Zheng, S.; Li, B. A novel image steganography method via deep convolutional generative adversarial networks. *IEEE Access* **2018**, *6*, 38303–38314. [CrossRef]
2. Cao, Y.; Zhou, Z.; Chakraborty, C.; Wang, M.; Wu, Q.J.; Sun, X.; Yu, K. Generative steganography based on long readable text generation. *IEEE Trans. Comput. Soc. Syst.* **2022**, *11*, 4584–4594. [CrossRef]
3. Liu, J.; Li, Z.; Jiang, X.; Zhang, Z. A high-performance CNN-applied HEVC steganography based on diamond-coded PU partition modes. *IEEE Trans. Multimed.* **2021**, *24*, 2084–2097. [CrossRef]
4. Wang, J.; Jia, X.; Kang, X.; Shi, Y.Q. A cover selection HEVC video steganography based on intra prediction mode. *IEEE Access* **2019**, *7*, 119393–119402. [CrossRef]
5. Mielikainen, J. LSB matching revisited. *IEEE Signal Process. Lett.* **2006**, *13*, 285–287. [CrossRef]
6. Liao, X.; Yu, Y.; Li, B.; Li, Z.; Qin, Z. A new payload partition strategy in color image steganography. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *30*, 685–696. [CrossRef]
7. Su, W.; Ni, J.; Hu, X.; Fridrich, J. Image steganography with symmetric embedding using Gaussian Markov random field model. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *31*, 1001–1015. [CrossRef]
8. Yang, J.; Ruan, D.; Huang, J.; Kang, X.; Shi, Y.Q. An embedding cost learning framework using GAN. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 839–851. [CrossRef]
9. Zhang, X.; Peng, F.; Long, M. Robust coverless image steganography based on DCT and LDA topic classification. *IEEE Trans. Multimed.* **2018**, *20*, 3223–3238. [CrossRef]
10. Chen, X.; Zhang, Z.; Qiu, A.; Xia, Z.; Xiong, N.N. Novel coverless steganography method based on image selection and StarGAN. *IEEE Trans. Netw. Sci. Eng.* **2020**, *9*, 219–230. [CrossRef]
11. Yang, Z.; Chen, K.; Zeng, K.; Zhang, W.; Yu, N. Provably secure robust image steganography. *IEEE Trans. Multimed.* **2023**, *26*, 5040–5053. [CrossRef]
12. Peng, F.; Chen, G.; Long, M. A robust coverless steganography based on generative adversarial networks and gradient descent approximation. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 5817–5829. [CrossRef]
13. Cao, Y.; Zhou, Z.; Wu, Q.J.; Yuan, C.; Sun, X. Coverless information hiding based on the generation of anime characters. *EURASIP J. Image Video Process.* **2020**, *2020*, 1–15. [CrossRef]
14. Wen, W.; Huang, H.; Qi, S.; Zhang, Y.; Fang, Y. Joint Coverless Steganography and Image Transformation for Covert Communication of Secret Messages. *IEEE Trans. Netw. Sci. Eng.* **2024**, *11*, 2951–2962. [CrossRef]
15. Zhou, Z.; Su, Y.; Li, J.; Yu, K.; Wu, Q.J.; Fu, Z.; Shi, Y. Secret-to-image reversible transformation for generative steganography. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 4118–4134. [CrossRef]
16. Zhou, Z.; Dong, X.; Meng, R.; Wang, M.; Yan, H.; Yu, K.; Choo, K.K. Generative steganography via auto-generation of semantic object contours. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 2751–2765. [CrossRef]
17. Cao, Y.; Ge, W.; Yuan, C.; Wang, Q. Generative Image Steganography via Encoding Pose Keypoints. *Appl. Sci.* **2024**, *15*, 58. [CrossRef]
18. Li, C.; Liu, H.; Fan, Z.; Li, W.; Liu, Y.; Pan, P.; Yuan, Y. Gaussianstego: A generalizable stenography pipeline for generative 3d gaussians splatting. *arXiv* **2024**, arXiv:2407.01301.
19. Su, P.C.; Cheng, Y.H.; Kuo, T.Y. JSN: Design and Analysis of JPEG Steganography Network. *Electronics* **2024**, *13*, 4821. [CrossRef]
20. Koptyra, K.; Ogiela, M.R. Steganography in QR Codes—Information Hiding with Suboptimal Segmentation. *Electronics* **2024**, *13*, 2658. [CrossRef]
21. Miranda, J.D.; Parada, D.J. LSB steganography detection in monochromatic still images using artificial neural networks. *Multimed. Tools Appl.* **2022**, *81*, 785–805. [CrossRef]
22. Kingma, D.P.; Dhariwal, P. Glow: Generative flow with invertible 1×1 convolutions. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; Volume 31.
23. Zhang, X. Behavior steganography in social network. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceeding of the Twelfth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kaohsiung, Taiwan, 21–23 November 2016*; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; Volume 1.
24. Gao, P.; Chai, P.; Lang, J. Behavior Steganography in Social Networks Based on 0–1 Knapsack Algorithm. *Acta Electron. Sin.* **2022**, *50*, 753–758.

25. Zhou, Z.; Su, Y.; Zhang, Y.; Xia, Z.; Du, S.; Gupta, B.B.; Qi, L. Coverless information hiding based on probability graph learning for secure communication in IOT environment. *IEEE Internet Things J.* **2021**, *9*, 9332–9341. [CrossRef]
26. Mahato, S.; Yadav, D.K.; Khan, D.A. A minesweeper game-based steganography scheme. *J. Inf. Secur. Appl.* **2017**, *32*, 1–14. [CrossRef]
27. Ou, Z.H.; Chen, L.H. A steganographic method based on tetris games. *Inf. Sci.* **2014**, *276*, 343–353. [CrossRef]
28. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
29. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef]
30. Kocsis, L.; Szepesvári, C. Bandit based monte-carlo planning. In *European Conference on Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 282–293.
31. Xu, G.; Wu, H.Z.; Shi, Y.Q. Structural design of convolutional neural networks for steganalysis. *IEEE Signal Process. Lett.* **2016**, *23*, 708–712. [CrossRef]
32. Ye, J.; Ni, J.; Yi, Y. Deep learning hierarchical representations for image steganalysis. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2545–2557. [CrossRef]
33. Huang, R.; Lian, C.; Dai, Z.; Li, Z.; Ma, Z. A novel hybrid image synthesis-mapping framework for steganography without embedding. *IEEE Access* **2023**, *11*, 113176–113188. [CrossRef]
34. Jing, J.; Deng, X.; Xu, M.; Wang, J.; Guan, Z. HiNet: Deep Image Hiding by Invertible Network. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 4713–4722.
35. Li, J.; Niu, K.; Liao, L.; Wang, L.; Liu, J.; Lei, Y.; Zhang, M. A generative steganography method based on WGAN-GP. In *Artificial Intelligence and Security: Proceedings of the 6th International Conference, ICAIS 2020, Hohhot, China, 17–20 July 2020*; Part I 6; Springer: Singapore, 2020; pp. 386–397.