electronics

Article

Efficiently Mastering the Game of NoGo with Deep Reinforcement Learning Supported by Domain Knowledge

Yifan Gao [1,*,†] and Lezhou Wu [2,†]

[1] College of Medicine and Biological Information Engineering, Northeastern University, Liaoning 110819, China
[2] College of Information Science and Engineering, Northeastern University, Liaoning 110819, China; 20184005@stu.neu.edu.cn
* Correspondence: yifangao@stumail.neu.edu.cn
† These authors contributed equally to this work.

Abstract: Computer games have been regarded as an important field of artificial intelligence (AI) for a long time. The AlphaZero structure has been successful in the game of Go, beating the top professional human players and becoming the baseline method in computer games. However, the AlphaZero training process requires tremendous computing resources, imposing additional difficulties for the AlphaZero-based AI. In this paper, we propose NoGoZero+ to improve the AlphaZero process and apply it to a game similar to Go, NoGo. NoGoZero+ employs several innovative features to improve training speed and performance, and most improvement strategies can be transferred to other nonspecific areas. This paper compares it with the original AlphaZero process, and results show that NoGoZero+ increases the training speed to about six times that of the original AlphaZero process. Moreover, in the experiment, our agent beat the original AlphaZero agent with a score of 81:19 after only being trained by 20,000 self-play games' data (small in quantity compared with 120,000 self-play games' data consumed by the original AlphaZero). The NoGo game program based on NoGoZero+ was the runner-up in the 2020 China Computer Game Championship (CCGC) with limited resources, defeating many AlphaZero-based programs. Our code, pretrained models, and self-play datasets are publicly available. The ultimate goal of this paper is to provide exploratory insights and mature auxiliary tools to enable AI researchers and computer-game communities to study, test, and improve these promising state-of-the-art methods at a much lower cost of computing resources.

Keywords: artificial intelligence; deep learning; AlphaZero; NoGo games; reinforcement learning

Citation: Gao, Y.; Wu, L. Efficiently Mastering the Game of NoGo with Deep Reinforcement Learning Supported by Domain Knowledge. *Electronics* **2021**, *10*, 1533. https://doi.org/10.3390/electronics10131533

Academic Editor: Amir Mosavi

Received: 31 May 2021
Accepted: 23 June 2021
Published: 24 June 2021

1. Introduction

The successive appearance of AlphaGo [1], AlphaGo Zero [2], and AlphaZero [3], achieving remarkable performance in one of the most complex games, Go, demonstrate the capabilities of deep reinforcement learning.

In 2017, Deepmind's AlphaGo Zero showed the possibility for computers to achieve superhuman performance in Go without relying on human knowledge or pre-existing data. Subsequently, AlphaZero made outstanding achievements in chess and shogi. However, a large amount of computational resources was required. Deepmind ran the training progress for Go for several days with 5000 TPUs, while Facebook's ELF OpenGo used 2000 V100 GPUs to achieve the top level of performance [4]. Therefore, it is an important and meaningful research direction to improve AlphaZero with limited computational resources.

This paper introduces several methods to speed up the training process and improve the final performance of the original AlphaGo Zero model (pipeline shown in Figure 1). The reinforced model is called NoGoZero+. Although NoGoZero+ uses some domain-specific features and optimization methods, it still starts from a random policy without using external strategic knowledge or existing data. Additionally, techniques used in

NoGoZero+ can be transferred to other nonspecific domains. By comparing the training process under the same condition in NoGo, the training efficiency of NoGoZero+ was at least six times that of the original AlphaZero.

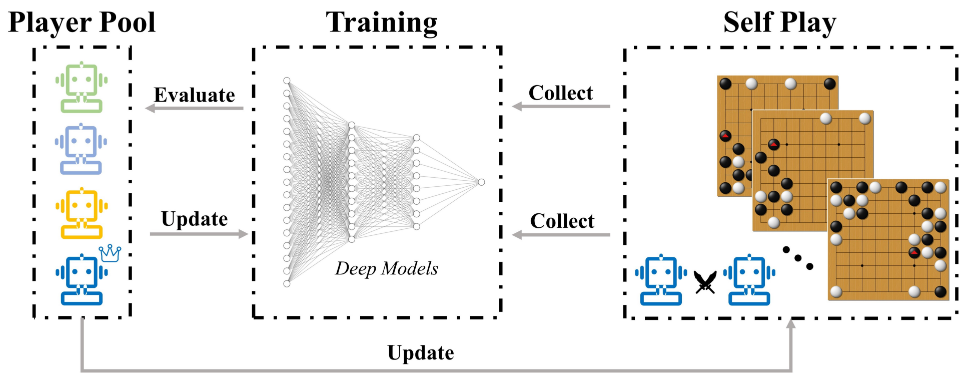

Figure 1. AlphaZero's pipeline. Self-play games' data are continuously generated and collected to train deep neural networks. After each round of training, the new model is compared with the previous model. If the new model defeats the previous model, the training process continues. Otherwise, the previous training result is discarded and the previous step is restarted.

NoGo is a new kind of board game that originated in 2005 [5]. Different than traditional Chinese board game Go, it forbids players from capturing stones. Once a player has no choice but to capture the counterpart's stones, the player loses. The formal rules of NoGo are shown below [6]:

- Board size is 9×9.
- Black goes first, and both sides take turn in moving in the board. A stone cannot be moved once the location is chosen.
- The goal of both sides is occupying areas instead of capturing counterpart's stones.
- One side loses if it captures the other side's stones or it suicides (deliberately makes its own stone to be captured by their counterpart).

Because of NoGo's novel rules and extremely limited background studies, NoGo does not have a mature strategy. This paper creates a precedent of successfully applying reinforcement learning to NoGo.

The main contributions are summarized as follows:

First, the paper proposes methods that can speed up the training progress and improve the final performance. These methods can be either directly or indirectly applied to similar AlphaZero training processes or general reinforcement-learning processes.

Second, the authors applied NoGoZero+ to the NoGo game and obtained better results than those achieved by the original AlphaZero under the condition of limited resources. This result shows the efficiency gap between AlphaZero's general methods and indicates the existence of better methods under specific conditions.

Third, to help research in this field, we provide the source code used to train the model, some pretrained high-level models, and a comprehensive self-play dataset that contains about 160,000 self-play games (available online: https://github.com/yifangao18/NoGoZero (accessed on 20 June 2021)).

The rest of this paper is organized as follows. In Section 2, we present some closely related works. The paper summarizes the basic architecture in Section 3 and describes the proposed techniques. In Section 4, we introduce experimental settings, criteria, and details. Section 5 reports the result and reviews the performance of NoGoZero+ in the competition. Section 6 discusses the results obtained with our approach. Lastly, we conclude our work and plan future works in Section 7.

2. Related Work

In this section, we first go over the AlphaGo family, which first introduced deep reinforcement learning (DRL) to board games with a large searching space, and some famous board game AI based on AlphaGo-like methods. Then, we introduce early work on the game of NoGo. Most of them achieved great success in the early years.

2.1. State of the Art in Board Game AI

The AlphaGo family, including AlphaGo, AlphaGo Zero, and AlphaZero [1–3], have had great success in many complex board games such as Go and chess. Unprecedentedly, AlphaGo family introduced DRL to board games, and DRL-based methods attracted many researchers' attention when AlphaGo beat top human player Lee Sedol. In particular, AlphaGo Zero successfully trains a Go AI from scratch using only the rules of the game because of the successful combination between Monte Carlo tree search (MCTS) [7,8] and deep neural network. Furthermore, the so-called zero-learning method used by AlphaZero is a more general method that can be used both in board games and other, more practical areas.

However, the large amount of computational resources consumed by the training process of AlphaZero and the relatively immature network structures encouraged many researchers to look into improved methods based on AlphaZero. An open-source project called ELF OpenGo [4] reached superhuman level in the game of Go after two weeks of training on 2000 GPUs (a relatively small number compared with the 5000 TPUs used by AlphaZero). KataGo [9], a reimplementation of AlphaGo Zero, improved the learning process in many ways, including using different optimization approaches to have more data about the value in a shorter period of time and using additional training targets to speed up the training process. Leela Zero [10], which is an open-source program trained with GPUs donated by a community of contributors, mastered Go and chess. Morandin et al. [11] proposed a sensible artificial intelligence (SAI) that plays Go to overcome the problem that AIs cannot target their margin of victory, and this is the common problem shared by most of the famous early-game AIs based on AlphaZero. Thus, the SAI successfully overcame the negative consequences: AIs often win by a small margin, cannot be used with komi 6.5, and show lousy play in handicap games.

2.2. NoGo AI Research

NoGo, as opposed to the ancient game of Go, is becoming the new favorite in the game AI community because of its relatively easy rules and lower computational resource requirements. Some early studies on NoGo AI achieved great results. Lee et al. [12] proposed an approach using ontologies, evolutionary computation, fuzzy logic, and fuzzy markup language combined with a genetic-algorithm-based system. Their NoGo AI could analyze the situation of the current board and play the next move to an inferred good-move position. Sun et al. [13] put forward a static-evaluation method to accurately estimate the value of each state of NoGo. Sun et al. [14] successfully used an improved pattern-matching algorithm to find out the best move in the game of NoGo.

As far as we know, there are few studies about the combination of DRL or zero-learning and the game of NoGo. Although former NoGo AI combined with traditional methods achieved relatively good performance, the AlphaGo family showed that there is a great performance gap between traditional methods and the DRL method. As a result, the implementation of DRL and zero-learning method in the game of NoGo is necessary.

3. Methods

In this section, we introduce methods that we used to build up NoGoZero+. We first go over the basic architecture of NoGoZero+. Then, we introduce the main novel techniques that we used to improve the training process and the final performance of NoGoZero+.

3.1. Basic Architecture

While NoGoZero+ has various novel details and improvements, it has a similar basic architecture to that of AlphaZero. We basically followed the parameters in [9] because this study provides complete parameter information compared to other AlphaZero implementations.

MCTS is the core searching method of AlphaZero, which is guided by a neural network. NoGoZero+ uses it to play the game against itself to generate training data. It also uses a variant of PUCT [15] to balance exploration and exploitation, and c_{puct} is a constant that determines the importance of both. When c_{puct} is small, MCTS tends to exploit game states with high value. Conversely, when c_{puct} is large, MCTS tends to explore unknown move locations, and the exploration is guided by the policy prior outputs by the neural network. With playouts repeating, the searching process continues, and the search tree grows. The process of playouts starts from the root and goes down the tree, and each node n selects child c with the largest PUCT(c) value:

$$PUCT(c) = V(c) + c_{puct}P(c)\frac{\sqrt{\sum_{c'} N(c')}}{1+N(c)} \tag{1}$$

where *V(c)* represents the average predicted score of all nodes in *c*'s subtree, *P(c)* represents the policy prior of *c* from the neural network, *N(c)* represents the number of playouts that are used to go through child *c*, and we set $c_{puct} = 1.1$. The Iteration of MCTS is terminated after a certain amount, and generates new policies on the basis of the visit frequency of their child nodes in the tree.

NoGoZero+ adds noise to the policy prior at the root to encourage exploration:

$$P(c) = 0.75P_{raw}(c) + 0.25\eta \tag{2}$$

where η is a draw from Dirichlet distribution on legal moves with parameter $\alpha = 0.03 * 9^2/N$, where N is the number of legal moves, and $P_{raw}(c)$ is the policy prior at the root.

The board of NoGo, unlike the traditional 19×19 Go board, has a size of 9×9, so the 9^2 part of parameter α corresponds to the NoGo board size. NoGoZero+ also applies a softmax temperature at the root of 1.03 to improve policy convergence stability according to [11].

A convolutional residual network [16] with preactivation is used to guide the search. The residual network has a trunk of b residual blocks with c channels. However, after various improvements, our network structure was very different compared to that of AlphaZero; see Figure 2. Furthermore, in the primary stage of training, inspired by curriculum learning [17], we designed a method called network curriculum learning to speed up the training progress. NoGoZero+ began with a relatively small residual network and gradually improved the size of the network when it converged under the condition of using the earlier residual network.

3.2. Techniques in NoGoZero+

3.2.1. Global Attention Residual Block

In general, the AlphaZero-based model has a strong ability to capture the local information of the board. However, due to the limited perceptual radius of the classical convolution layer, global strategy prediction remains challenging. Some findings in previous studies on board games, such as the fatal 'ladder failure' that commonly occurs in Go, support this view [4]. Therefore, we began exploring the attention-based structure. As a popular design of computer vision, the attention mechanism can help models to globally pay more attention to important information.

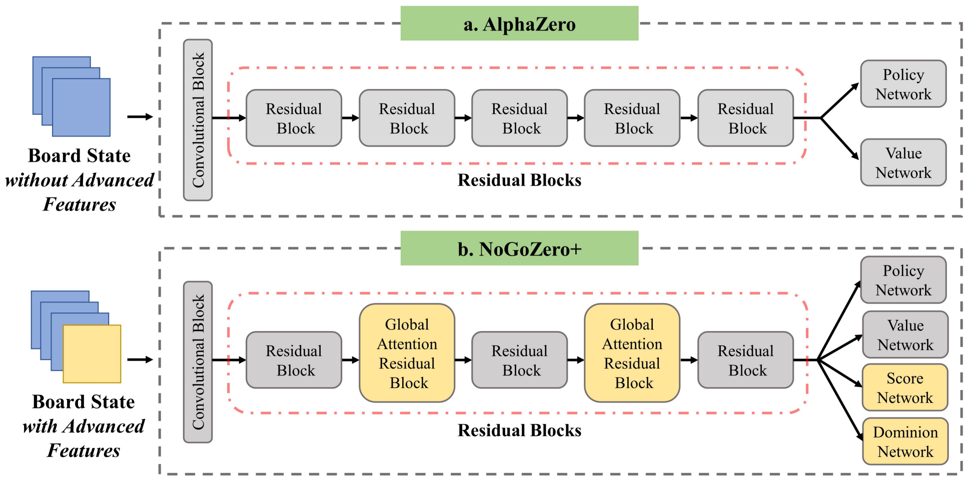

Figure 2. Comparison of network architecture of AlphaZero and NoGoZero+ (5 residual blocks).

In this section, we propose the global attention residual block (GARB), a novel attention-based structure based on the global pooling layer, as shown in Figure 3. The global pooling layer was proposed in previous research on the game of Go [9]. The layer enables the convolutional layers to condition in a global context, which can be hard or impossible for convolutional layers with limited perceptual radius. NoGoZero+ replaces parts of the ordinary convolutional layers with the global pooling structure (GPS) to improve the neural network's ability of synthesizing the global context.

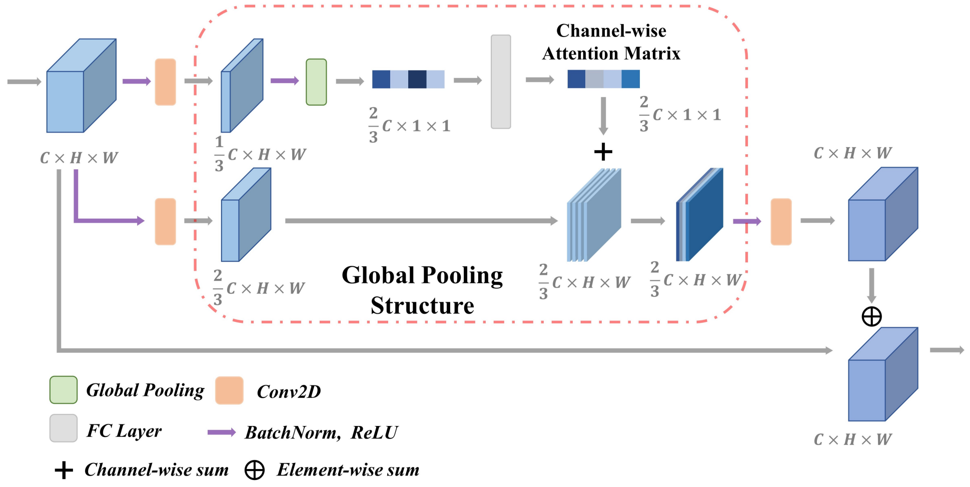

Figure 3. Proposed GARB. The structure globally aggregates values of one set of channels to bias another set of channels, potentially providing the final output with information on the global context in NoGoZero+.

Given a set of c channels, the global pooling layer computes the mean of each channel and the maximum of each channel. The whole process outputs a total of $2c$ values. The global pooling layer is a part of the GPS. Given input $X \in R^{C_1 \times H \times W}$ and $G \in R^{C_2 \times H \times W}$ (C_1 and C_2 represent the channel, H and W represent the height and width respectively), the GPS includes the following components:

- Batch-normalization layer and rectified linear unit (ReLU) activation function applied to G, output shape $C_2 \times H \times W$.
- Global pooling layer applied to G, output shape $2C_2$.
- Fully connected (FC) layer to G, output shape C_1.
- Channelwise sum with X, output shape $C_1 \times H \times W$.

Previous studies showed that avoiding dimensionality reduction in the FC layer is crucial for learning channel attention [18]. Therefore, we set the value of C_1 to twice that of C_2 to avoid performance degradation caused by dimensional changes. GPS was inserted into a standard residual block and eventually constituted GARB. Given input $X_{in} \in R^{C\times H\times W}$ of GARB, two independent convolutional layers decomposed the input into X with channel dimension number $\frac{2C}{3}$, and G with channel dimension number $\frac{C}{3}$ before sending it to GPS.

Moreover, GARB is more focused on extracting long-range dependencies, and it is insufficient for local-information extraction compared to the standard residual block. In order to supplement local information, GARB alternates between two configurations in consecutive residual blocks, as illustrated in Figure 2b.

3.2.2. Multitask Learning

AlphaZero has both a policy head and a value head at the end of its deep neural network. They contribute to the policy network and value network, respectively. The policy head predicts potential good moves, and the value head predicts the final result of the game.The deep neural network outputs policy and value on the basis of the current state of the board. Two parallel networks can be regarded as two training tasks for the neural network.

Such a multitask learning method [19] has had great success in training AlphaZero. NoGoZero+ follows and expands the idea. The paper adds two other output heads, dominion head and score head, which contribute to the two extra tasks (shown in Figure 4a). In NoGo, dominion head predicts the ownership of each location on the board. The ownership indicates the key locations that heavily impact the final result. In both computer games and real world, the final result is suggested both by outputs of a prediction network or noisy 1 and -1 (win and loss), and by more details observed during the game. The neural network can have more insight into the cause of the final result, including the true gap between the winner and loser in a playout round.

Our proposed dominion head architecture is shown in Figure 4b. It contains a convolutional block, a global pooling convolutional block, and an output layer. The convolutional block includes a batch-normalization layer, a ReLU activation layer, and a 3×3 convolutional layer. The global pooling convolutional block includes the GPS, a batch-normalization layer, a ReLU activation layer, and a 1×1 convolutional layer with 2 filters.

We assumed input feature map $x \in R^{C\times H\times W}$, where C, H, and W are channel, height, and width. First, the convolutional block is applied to x, output with the same size as that of x. Then, the output of the convolutional block is passed through the global pooling convolutional block, and the number of channel dimensions of x is reduced to 2. Lastly, we apply a FC layer with sigmoid activation in the output layer that provides dominion output $y_d \in R^{H\times W}$.

Our proposed score-head architecture is shown in Figure 4c. The network structure of the score head is similar to that of the dominion head, with two differences. The first is that the 1×1 convolutional layer of the global pooling convolutional block has only one filter, so the output feature map is 1 in the channel dimension. The second is that the output layer of the score head contains a FC layer with tanh activation, outputting a scalar in the range of $[-1, 1]$. Lastly, we multiply it by the score factor to reflect the score gap between the two players. In NoGo, since most score gaps are within 5 points, the score factor was set to 5, and a larger score gap is regarded as 5 points once appearing. Therefore, output ys is a scalar in the range of $[-5, 5]$.

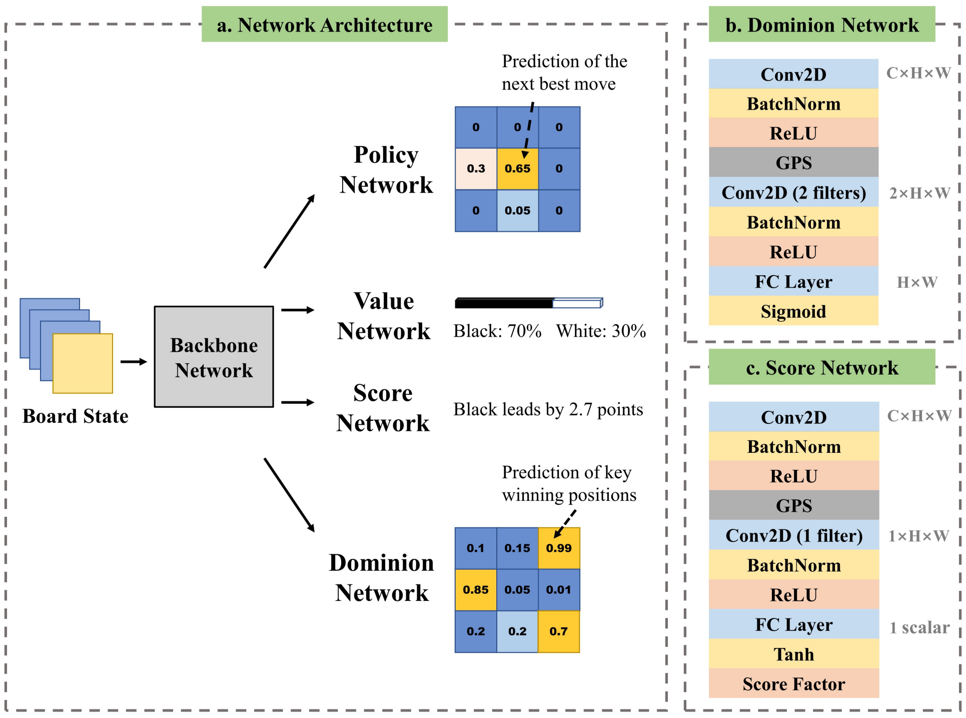

Figure 4. (**a**) Multitask learning network structure. (**b**) Illustration of dominion head. (**c**) Illustration of the score head. Except for the policy head and value head, which are also used in AlphaZero, we added score head and dominion head to evaluate the gap of the performance of two players and the key winning position, respectively. The two additional heads can effectively help the NoGoZero+ agent have insight into the game state and make full use of self-play data because of the more detailed information.

3.2.3. Network Curriculum Learning

Curriculum learning is an important method in reinforcement learning [20,21]. To prevent the agent from being stuck at the early stage of training because of the hard initial task or having dissatisfactory performance at the end of training because of oversimplified tasks, we carried out curriculum learning by imitating the learning process of humans and animals and gradually improve the model trough progressive samples and knowledge.

In this paper, the idea of curriculum learning was extended to the neural network and is called ‘network curriculum learning’. A ahallower and narrower structure leads to a network with less complexity, but it converges more quickly. While the deeper and wider network with more complexity is usually harder to be trained, network curriculum learning tries to balance complexity and convergence rate. During NoGoZero+ training, the process starts with a simpler network with fewer parameters. Self-play and training data are generated by the shallow network, and used to train a deeper and more complex network (and the simpler network itself), while the deeper network itself does not generate new data. As soon as the simpler network meets its bottleneck, which means that this network has converged, the whole training process is transferred to a larger network. The process is repeated until the size and the performance of the network satisfy our needs.

Figure 5 shows the course of network curriculum learning. The whole process of training the NoGoZero+ agent can be separated into three stages, and the number of residual blocks (b in Figure 5) grew from 5 to 20, while the number of channels (c in Figure 5) of each residual block grew from 32 to 128. With the help of network curriculum learning, the agent avoids the stagnation of training in the early stage and saves much training time.

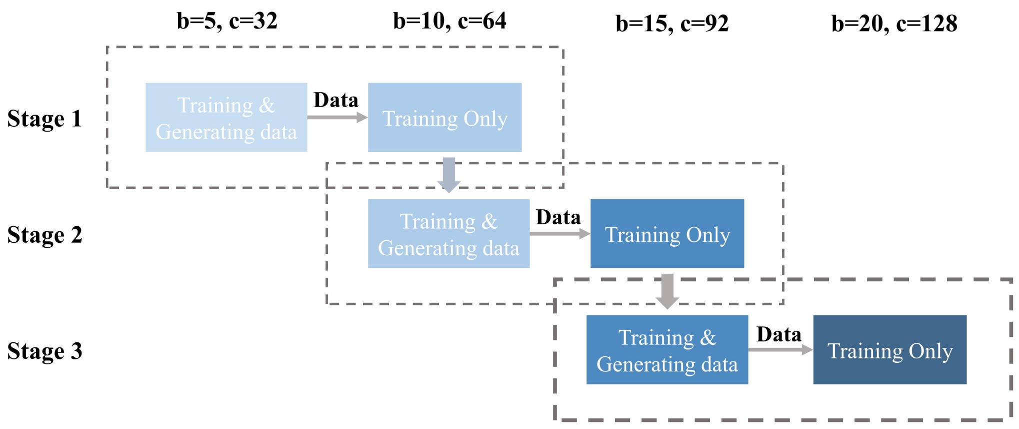

Figure 5. Process of network curriculum learning. Data generated by the shallower network are used to train both the shallow network itself and a deeper network. The whole process contains three stages, and the network with b = 20, c = 128 was extracted as the final network that is used in the agent.

3.2.4. Advanced Specific Features

AlphaZero provides a universal method to train game agents. The method performs perfectly on chess, shogi, and Go. However, a sea of computational resources that make the universal method work. With limited resources, it is important for us to add some advanced specific features to speed up the training process and make full use of the obtained data from every game. Like other machine-learning methods [22], reinforcement learning also attaches much importance to well-designed specific features, which can make a great difference to the speed of training process, as well as the final performance of agents.

Similar to imitation learning, some domain-specific professional knowledge guides the deep model to converge faster during the initial stage. Besides current and historical steps, which are also the features used by AlphaZero, NoGoZero+ adds another kind of advanced feature, liberty, which is also called 'qi'. Four input layers are added on the basis of the input layers of the network to show the features of liberty:

- Liberty locations belonging to our stones that have only one liberty.
- Liberty locations belonging to our stones that have two liberties.
- Liberty locations belonging to the counterpart's stones that have only one liberty.
- Liberty locations belonging to the counterpart's stones that have two liberties.

The four above features are (1) 'half-dead', (2) 'near-dead', (3) 'half-kill', and (4) 'near-kill', respectively. One-hot encoding is used to represent the four advanced features. The corresponding location in the input layer is set to 1 once a specific feature appears. The corresponding features are schematically shown in Figure 6.

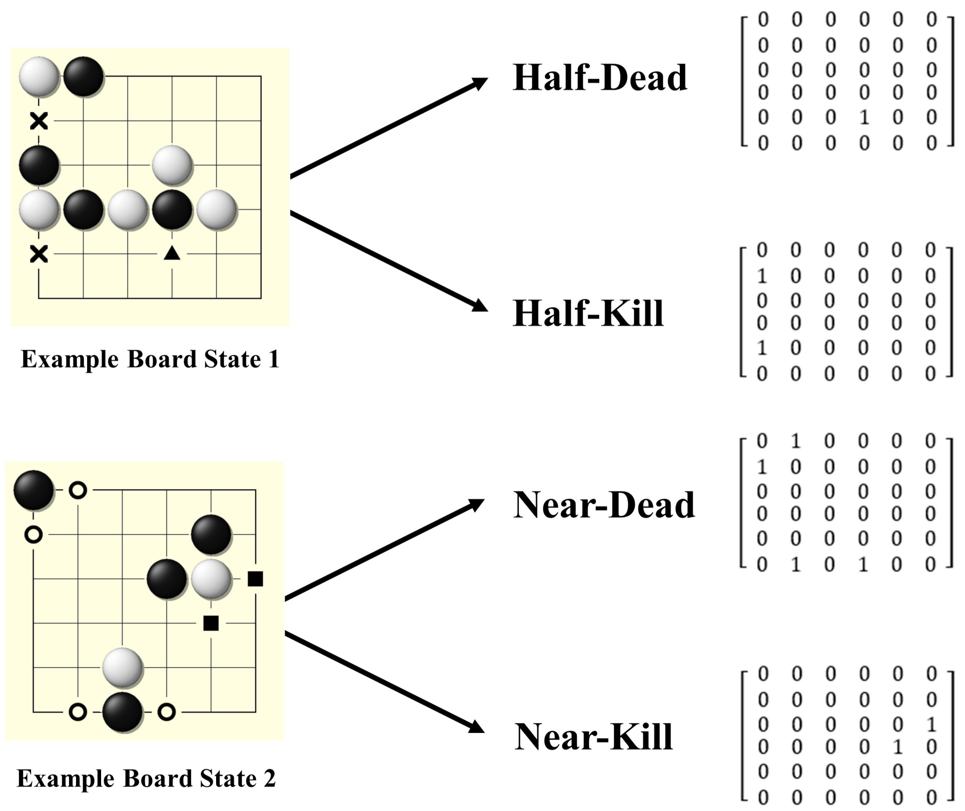

Figure 6. Demonstration of advanced features. Location of existing features is marked with one-hot encoding in the corresponding input layer. Four additional layers are added to the input layer set of the original AlphaZero and provide the NoGoZero+ agent with additional advanced information about the board state. Results in the next section show that such modification can obviously improve training speed and final performance.

4. Experiments

In the two following sections, we use some control tests to verify the improvements our techniques bring to the basic AlphaZero method, and use ablation experiments to examine the contributions of each approach used in NoGoZero+. In this section, we introduce the experimental settings, evaluation criteria, and training parameters to detail the background and process of our experiments.

4.1. Experimental Settings

To look into the behavior of NoGoZero+ and demonstrate the effects of the unique techniques introduced in the paper, two experiments were conducted.

First, we designed an experiment to show the significant speedups attributing to adding extra training methods to the original AlphaZero theory by comparing three models' training processes on the NoGo:

- Training process of the original AlphaZero.
- Training process of AlphaZero with network-curriculum-learning technique.
- Training process of NoGoZero+.

Second, to explore the effects of every single technique, an ablation experiment was conducted. To more clearly demonstrate the contributions of each technique to the final result, all agents in the ablation experiment did not use the network-curriculum-learning method.

4.2. Evaluation Criteria

The Elo model [23] is an effective way to measure the level of game agents, widely used to evaluate the performance of agents in, for example, chess, Go, basketball, and football. Assuming that the Elo ratings of Agents A and B are R_A and R_B, respectively, the expected winning probability of A versus B is:

$$E_A = \frac{1}{1 + 10^{\frac{R_B - R_A}{400}}} \tag{3}$$

When Player A's true score S_A is different from its expected winning probability (1 for win, 0 for loss, 0.5 for tie. There is no tie in NoGo.), its Elo rating should be adjusted as

$$R_A' = R_A + \alpha(S_A - E_A) \tag{4}$$

where R'_A is the Elo rating of Player A after adjustment, and α is a weight that is often set to 16 in master tournaments (α = 16 is also used in our experiment).

4.3. Training Parameters and Details

We set stochastic gradient descent (SGD) as the optimizer with a learning rate equal to 0.001 and momentum equal to 0.8. Our experiments were performed on an NVIDIA Tesla V100 GPU (with 32 GB GPU memory). The deep-learning models were implemented using Pytorch. In training, the data-augmentation technique that we used includes all eight reflections and rotations for each position. The entire training process took about 100 h, and most of the resources were consumed in self-play. Multithreading self-play is recommended to make full use of GPU and CPU resources.

Considering the four output heads of the improved network structure, the loss function is the sum of the five following kinds of losses.

- Policy Loss

$$-\sum_m \pi(m) \log(\hat{\pi}(m)) \tag{5}$$

  where m is the set of the rest legal moves, π is the target policy derived from the playouts of the MCTS search, and $\hat{\pi}$ is the neural network's prediction of policy π.

- Value Loss

$$-c_g \sum_r z(r) \log(\hat{z}(r)) \tag{6}$$

  where r is the final result for the current player (($r \in \{win,\ lost\}$), z is a one-hot encoding function of r, $\hat{z}$ is the neural network's prediction of the final result. c_g = 1.5 is a scaling constant.

- Dominion Loss

$$-c_d \sum_m \gamma(m) \log(\hat{\gamma}(m)) + (1 - \gamma(m)) \log(1 - \hat{\gamma}(m)) \tag{7}$$

  where m is the set of the moves in the board, γ is the final result for the current player ($r \in \{dominion,\ not\ dominion\}$), $\hat{z}$ is the neural network's prediction of γ. c_d = 0.5 is a scaling constant.

- Score Loss

$$c_s(\hat{v} - v)^2 \tag{8}$$

  where v is the scalar of score difference, $\hat{v}$ is the prediction of the final result. c_s = 0.25 is a scaling constant.

- L2 Penalty

$$c_{L2}\|\theta\|^2 \tag{9}$$

  where c_{L2} = 10^{-5}, so as to prevent the network from overfitting due to the relatively deep neural network structure.

5. Results

In this section, the experiment results are shown and discussed. Then, some interesting playouts are displayed and explained to demonstrate the intelligence of NoGoZero+. Lastly, we introduce the performance of NoGoZero+ in the competition and summarize the possible reasons for not winning first place.

5.1. Experiment Results

The results of the first experiment are shown in Table 1. AlphaZero's Elo rating reached 2500 after 120,000 self-play games, while it only took NoGoZero+ 20,000 self-play games to reach an Elo rating of 2750, and NoGoZero+ defeated original AlphaZero with 81 wins and 19 losses in 100 playouts. The AlphaZero agent, with the help of the network-curriculum-learning method, had a similar learning curve to that of NoGoZero+. However, an obvious gap of Elo rating between the AlphaZero agent with network curriculum learning and complete NoGoZero+ indicated that other techniques that we use make a great difference in the behavior of the agent.

Table 1. Comparison of Elo ratings among original AlphaZero, AlphaZero with only network curriculum learning, and complete NoGoZero+. NCL, network curriculum learning. NoGoZero+ had advantages over the original AlphaZero in final performance (Elo rating) and training speed. Moreover, the obvious gap between complete NoGoZero+ and AlphaZero with only network curriculum learning indicated the supplementary power of other techniques. Blocks represent the number of residual blocks in the network.

Method	Blocks	NCL	Games	Elo Score
AlphaZero	20b		120k	2500
AlphaZero	5b	✓	3k	1800
AlphaZero	10b	✓	10k	2250
AlphaZero	20b	✓	20k	2350
NoGoZero+	5b	✓	3k	2300
NoGoZero+	10b	✓	10k	2625
NoGoZero+	20b	✓	20k	2750

The results of the second experiment (ablation experiments) are shown in Table 2. Comparing the final behavior of the agents with different parts weeded out, GARB had a relatively larger influence on the final performance (the Elo rating of which dropped from 2300 to 2045 without it), while multitask learning and advanced features closely impacted the performance of the agent. Adding advanced features was less influential because the main aim of adding advanced features is to help the agent make full use of detailed information obtained from one single playout to speed up the training process with relatively limited resources. Overall, every technique used in the training process of NoGoZero+ had a positive effect, which can effectively help to improve the Elo rating of the agent.

5.2. Strategies Learnt by the Agent

Unlike Go, which originated thousands of years ago and has been researched for quite a long time, NoGo is a new board game with few mature strategies. However, with the help of techniques shown in the previous sections, the NoGoZero+ agent developed a series of inspirational strategies. The paper demonstrates and discusses some distinct, impressive strategies shown by the NoGoZero+ agent in self-play games.

Table 2. Comparison of Elo ratings among original AlphaZero, AlphaZero with only network curriculum learning, AlphaZero using some of the techniques in this paper, and complete NoGoZero+. ASF, advanced specific features; MTL, multitask learning network structure; 5b, number of residual blocks in the network.

AlphaZero (5b)	GARB	MTL	ASF	Games	Elo Score
✓				3k	1800
✓	✓	✓		3k	2174 ± 47.62
✓	✓		✓	3k	2134 ± 52.24
✓		✓	✓	3k	2045 ± 87.39
✓	✓	✓	✓	3k	2300

5.2.1. Triangle Strategy

A triangle layout and a prismatic layout are two kinds of favorable layouts to players because the central location of such layouts cannot be occupied by a counterpart's stones. As a result, such layouts can greatly help at the end of the game when there are few locations to place a stone. At the beginning of a game, the agent tries to build as many triangle and prismatic layouts as possible, if not disturbed by the counterpart player (see Figure 7). Moreover, the agent sometimes manages to prevent its counterpart from successfully building a triangle or a prismatic layout after weighing advantages and disadvantages. Such behavior is called the 'triangle strategy'.

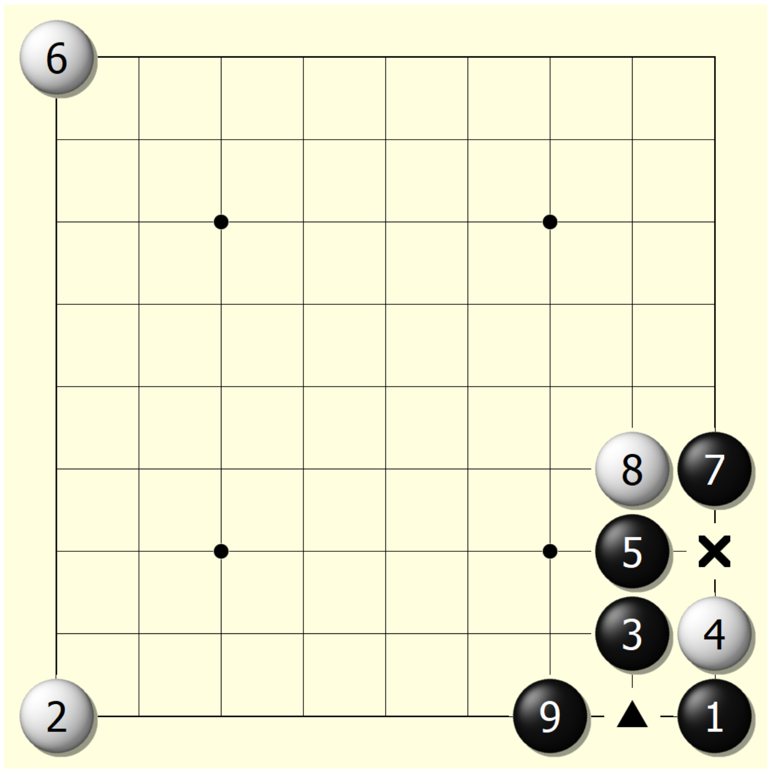

Figure 7. Demonstration of 'triangle strategy'. Black successfully built a triangle structure in the location marked by △, and white successfully prevented black from building triangle structure in Step 4 because neither black nor white could occupy the location marked by ×. Moreover, the white stone placed in Step 8 also tried to destroy a potential triangle structure that may have contained the black stones placed in Steps 5 and 7.

5.2.2. Predictive Strategy

With the progress of the NoGoZero+ agent, it becomes harder for white to win (the average win rate is 4.5) because black learns to make full use of the sente advantage.

A typical situation is: white tries hard to prevent black from using the triangle strategy and fails because black takes the lead in the order (Figure 8).

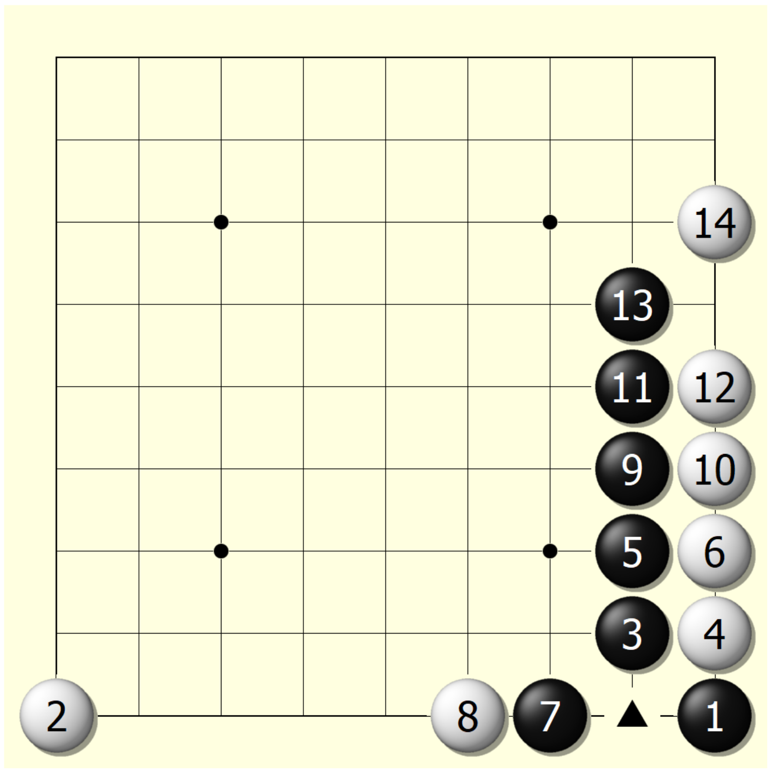

Figure 8. A typical situation where the white falls behind. Starting from Step 4, white tried hard to prevent black from using the triangle strategy, but black always took the lead. Although white made a wise choice in Step 14 and terminated the 'chasing game', black successfully built a triangle structure in Step 7. The whole process indicates that black has an obvious advantage over white when the NoGoZero+ agent is well-trained.

Since NoGo is considered to be a game where the black player is more likely to win, the white player needs to adopt a more aggressive strategy to win the game. We looked into games in which white won, and found that white seemed to gain foresight under certain conditions.

Figure 9 shows a part of a game where the white player won. Instead of chasing black's steps and trying hard to stop black in a relatively small area, the white player controlled the overall situation and successfully disturbed black's deployment on a larger scale.

Such a decision shows a sense of prediction, so the behavior is called 'predictive strategy'. The sense of the overall situation and the prediction should be attributed to the GARB, which gives the agent a high level of understanding of the whole situation, and multitask learning makes the agent capable of predicting the behavior of its counterpart.

Although the white player did not fully understand the predictive strategy in the experiment, it is impressive that the agent developed such a high-level tactic. Hopefully, after more self-play games, the agent can master predictive strategy or even other high-level strategies.

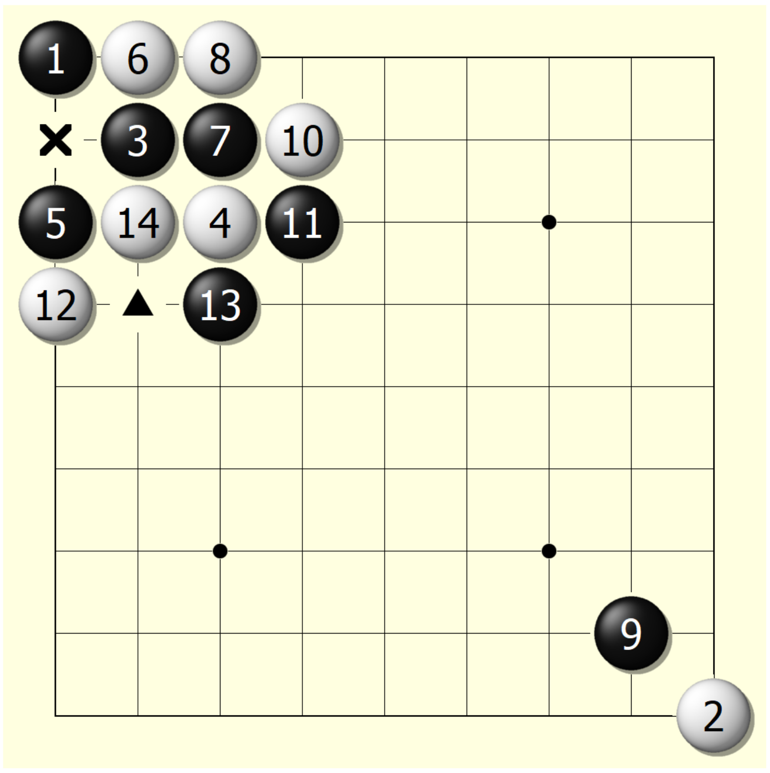

Figure 9. Demonstration of predictive strategy. When black tried to build a triangle structure in Step 3, instead of chasing black's step like the behavior shown in Figure 8, white took Step 4 and built a wider encirclement, thus rendering the triangle structure built by black (marked by ×) useless.

5.3. NoGoZero+ in CCGC

The China Computer Game Championship (CCGC) is the largest computer-game competition in China, held annually since 2006. In 2020, a total of 18 teams entered the national finals. Our team competed for the first time and unfortunately lost in the finals to the KnighTeam-NG (KNG) program developed by the Chongqing University of Technology. KNG has won consecutive championships in NoGo since 2014. As the code is not open-source, we cannot know what kind of technology KNG uses.

One lesson is that we unreasonably allocated the game time. NoGoZero+ spent much time in the first half of the game, and when the second half of the time was exhausted, some strange and unreasonable moves appeared before it lastly lost the game. In addition, our equipment for participating in the competition was not good enough. For each move, NoGoZero+ could only perform 1400–1800 MCTS on average, which led to worse model performance.

6. Discussion

Although the main battlefield of NoGoZero+ is still the game of NoGo, the techniques used by NoGoZero+ can be extended to other nonspecific areas to reduce resource consumption caused by large search spaces. First, GARB is an efficient attention mechanism that helps improve the network's ability to perceive global information. Second, the success of the multitask learning network structure proved that adding more output heads to neural networks can efficiently help the training process. Third, the network-curriculum-learning technique can help to balance the complexity and the rate of convergence of a deep neural network. Fourth, adding moderate domain-specific knowledge to the training process of neural networks can make full use of training data and reduce the training period, especially when computational resources are limited.

The game of NoGo is becoming a useful tool for many researchers who are interested in game AI but do not want to spend too much time learning the complex game rules to carry out studies. On the basis of the NoGoZero+ result, they can quickly develop a basic structure of

NoGo game AI and save time for further study. As a result, with the help of techniques used by NoGoZero+, the game AI community benefits from the lower research thresholds.

Moreover, AlphaZero-based methods mainly use convolutional neural networks (CNN) to capture board information, just like CNN captures information from pictures in computer-vision tasks. This means that many tricks used in computer vision can be transformed into AI training processes. The attention model, which is commonly used to select the most crucial information from pictures during computer-vision tasks, can be combined with the original CNN structure to improve efficiency in the usage of computational resources, and further enhance the final performance of the AI. This method is reasonable and needs further research because some positions are also more important than others are in board games.

7. Conclusions and Future Work

This paper presented NoGoZero+, which efficiently masters the game of NoGo on the basis of the improved AlphaZero algorithm. By using several techniques, the improvement is nearly cost-free. The methods enable a unified reinforcement-learning-based system to be trained from scratch to being a powerful agent in a few days. The techniques used in NoGoZero+ can easily be extended to other areas to reduce computational-resource consumption and improve training efficiency. Experiments showed that NoGoZero+ had six times better training speed and better performance than those of the original AlphaZero. This study highlights the tremendous potential of reducing computing resources in board-game AI.

Although the experiment results of this study are encouraging, there are still some limitations. For example, we were only awarded second place in the CCGC. The future of this study includes exploring more powerful neural-network architectures and more efficient sample-utilization methods to further improve performance. Our ultimate goal is to provide insights and additional tools for the community to explore large-scale deep-learning methods of computer games. Our code and data are public to help researchers in the game AI community.

Author Contributions: Conceptualization, Y.G. and L.W.; methodology, L.W. and Y.G.; validation, L.W. and Y.G.; resources, Y.G. and L.W.; data curation, Y.G. and L.W.; writing—original-draft preparation, Y.G. and L.W.; writing—review and editing, L.W. and Y.G.; supervision—L.W. and Y.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors thank Jiao Wang from the College of Information Science and Engineering, Northeastern University, China for their support in preparing this study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. 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]
2. 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] [PubMed]
3. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **2018**, *362*, 1140–1144. [CrossRef] [PubMed]
4. Tian, Y.; Ma, J.; Gong, Q.; Sengupta, S.; Chen, Z.; Pinkerton, J.; Zitnick, L. Elf OpenGo: An analysis and open reimplementation of AlphaZero. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 6244–6253.
5. Moore, J. NoGo History and Competitions. Available online: http://webdocs.cs.ualberta.ca/~mmueller/nogo/history.html (accessed on 24 June 2021).
6. CCGC Organizing Committee. University Computer Games Championship & National Computer Games Tournament. Available online: http://computergames.caai.cn/jsgz05.html (accessed on 24 June 2021).

7. Coulom, R. Efficient selectivity and backup operators in Monte-Carlo tree search. In Proceedings of the International Conference on Computers and Games, Turin, Italy, 29–31 May 2006; pp. 72–83.
8. Kocsis, L.; Szepesvári, C. Bandit based Monte-Carlo planning. In Proceedings of the European Conference on Machine Learning (ECML), Berlin, Germany, 18–22 September 2006; pp. 282–293.
9. Wu, D.J. Accelerating self-play learning in Go. *arXiv* **2019**, arXiv:1902.10565.
10. Pascutto, G.-C. Leela Zero. Available online: https://github.com/leela-zero/leela-zero (accessed on 24 June 2021).
11. Morandin, F.; Amato, G.; Gini, R.; Metta, C.; Parton, M.; Pascutto, G.C. SAI, a Sensible Artificial Intelligence that plays Go. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–8. [CrossRef]
12. Lee, C.S.; Wang, M.H.; Chen, Y.J.; Hagras, H.; Wu, M.J.; Teytaud, O. Genetic fuzzy markup language for game of NoGo. *Knowl. Based Syst.* **2012**, *34*, 64–80. [CrossRef]
13. Sun, Y.; Rao, G.; Sun, H.; Wei, Y. Research on static evaluation method for computer game of NoGo. In Proceedings of the 26th Chinese Control and Decision Conference, Changsha, China, 31 May–2 June 2014; pp. 3455–3459. [CrossRef]
14. Sun, Y.; Wang, Y.; Li, F. Pattern matching and Monte-Carlo simulation mechanism for the game of NoGo. In Proceedings of the IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, Hangzhou, China, 30 October 2012; Volume 1, pp. 61–64. [CrossRef]
15. Rosin, C.D. Multi-armed bandits with episode context. *Ann. Math. Artif. Intell.* **2011**, *61*, 203–230. [CrossRef]
16. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
17. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
18. Wang, Q.; Wu, B.; Zhu, P.; Li, P.; Zuo, W.; Hu, Q. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–18 June 2020.
19. Caruana, R. Multitask learning. *Mach. Learn.* **1997**, *28*, 41–75. [CrossRef]
20. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
21. Szepesvári, C. Algorithms for reinforcement learning. *Synth. Lect. Artif. Intell. Mach. Learn.* **2010**, *4*, 1–103. [CrossRef]
22. Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Prentice Hall: Hoboken, NJ, USA, 2002.
23. Elo, A. *The Rating of Chess Players, Past and Present*; Arco Publishing: London, UK, 1978.