electronics

MDPI

Article

New Hybrid Graph Convolution Neural Network with Applications in Game Strategy

Hanyue Xu [1], Kah Phooi Seng [1,2,3,*] and Li-Minn Ang [3]

[1] School of AI & Advanced Computing, Xi'an Jiaotong Liverpool University, Suzhou 215123, China; hanyue.xu19@student.xjtlu.edu.cn
[2] School of Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia
[3] School of Science, Technology and Engineering, University of Sunshine Coast, Petrie, QLD 4502, Australia; lang@usc.edu.au
* Correspondence: jasmine.seng@xjtlu.edu.cn or kahphooi.seng@qut.edu.au

Abstract: Deep convolutional neural networks (DCNNs) have enjoyed much success in many applications, such as computer vision, automated medical diagnosis, autonomous systems, etc. Another application of DCNNs is for game strategies, where the deep neural network architecture can be used to directly represent and learn strategies from expert players on different sides. Many game states can be expressed not only as a matrix data structure suitable for DCNN training but also as a graph data structure. Most of the available DCNN methods ignore the territory characteristics of both sides' positions based on the game rules. Therefore, in this paper, we propose a hybrid approach to the graph neural network to extract the features of the model of game-playing strategies and fuse it into a DCNN. As a graph learning model, graph convolutional networks (GCNs) provide a scheme by which to extract the features in a graph structure, which can better extract the features in the relationship between the game-playing strategies. We validate the work and design a hybrid network to integrate GCNs and DCNNs in the game of Go and show that on the KGS Go dataset, the performance of the hybrid model outperforms the traditional DCNN model. The hybrid model demonstrates a good performance in extracting the game strategy of Go.

Keywords: graph convolutional networks; deep convolutional neural networks; game strategies

Citation: Xu, H.; Seng, K.P.; Ang, L.-M. New Hybrid Graph Convolution Neural Network with Applications in Game Strategy. *Electronics* **2023**, *12*, 4020. https://doi.org/10.3390/electronics12194020

Academic Editor: Praveen Kumar Donta

Received: 2 September 2023
Revised: 21 September 2023
Accepted: 22 September 2023
Published: 24 September 2023

1. Introduction

In game theory, strategy is a plan of action developed by players that depends not only on their own actions but also on whatever options other players have in the environment. It can serve as a blueprint for the player's decisions, actions, and interactions in the game environment. In recent years, due to Google DeepMind's research in the field of strategy games such as Go, chess, and shogi [1–3], this has once again become a hot topic. To date, various decision-making algorithms for strategy games have been proposed. Based on the operating principle, the algorithm can be divided into two groups, i.e., traditional artificial intelligence (AI) algorithms and machine learning AI algorithms.

In the early days, most approaches were based on the human formulation of specific rules and logic for decision-making and inference, such as finite-state machine (FSM) [4], tree search [5], and utility-based AI [6]. However, these methods are not suitable for most strategy games because the specific use of algorithms depends on the application scenario and needs. To alleviate this problem, these methods have been modified in different decision games; for example, the Monte Carlo tree search algorithm solves the particularity of excessive search space in the game of Go [7]. While these traditional AI algorithms require low computing costs and are easy to design and tune, as the game scene becomes more complex and the number of players increases, the rules become more difficult to design.

Compared with traditional AI algorithms, machine learning AI algorithms can learn independently of existing player game data and have decision-making capabilities. Deep learning [8] and reinforcement learning [9] are machine learning AI algorithms commonly used in the field of gaming. Both methods require a large number of datasets for training. The difference is that deep learning aims to learn from the experience of existing experts and imitate their strategies, while reinforcement learning aims to strengthen performance through self-exploration and repeated practice with itself. Although machine learning AI algorithms can solve problems for complex scenarios, a single model cannot satisfy the evaluation of a global game. In the game of Go, the traditional heuristic evaluation function makes it difficult to evaluate the complexity of the situation accurately; thus, the reinforcement learning model may be affected.

In deep learning methods, deep convolutional neural networks (DCNNs) have enjoyed much success in many applications such as computer vision [10], automated medical diagnosis [11], autonomous systems [12], etc. Another application of DCNNs is for game strategies, where the deep neural network architecture can be used to directly represent and learn strategies from expert players on different sides. From the simplest strategy games, Wang et al. [13] utilized the CNN to simulate the Pac-Man game's decision-making process and explain each convolutional layer's internal logic. Sutskever and Nair [14] introduced the CNN model to Go, a game with high decision complexity, and the prediction of expert moves reached 36.9%. To improve decision-making performance, Clark and Storkey [15] proposed a DCNN model with multiple hidden layers, which combined features built manually based on Go rules to achieve better prediction results. In addition, an efficient hardware architecture was used to accelerate the DCNN models. Li et al. [16] proposed a hardware architecture that can accommodate different field-programmable gate arrays (FPGA) to speed up the DCNN model and balance processing speed and hardware resources.

Due to the diversity of strategy game rules, complex distribution space, and different numbers of players, the extraction of game gradient features is needed in order to satisfy all the gameplay of strategy games. As shown in Figure 1, space four chess, as a board strategy game, is different from chess and Go, being a game that can be played in three-dimensional space. The winning condition is that the pieces of the same color reach four consecutive placements. The chess pieces not only can achieve the winning states in the horizontal, vertical, and oblique directions of the plane but also in the oblique placement of the space (see Figure 1). Because the DCNN is designed for Euclidean data, it is difficult for such a complex distributed space strategy game to perform as well as it should. While some of the DCNN-based approaches work well in traditional strategy games to some extent, they tend to ignore the relevance of camps between gamers. For example, in the game of Go, the condition of victory is based on the size of the territory occupied by the player's stones. Therefore, the correlation between the areas occupied by each stone is significant for the overall evaluation of the board, which affects the player's next decision. However, DCNNs only focus on the information extraction at the image level and ignore the features of the dimension of correlation between adjacent fields.

Figure 1. The rules of space four chess play.

Many game states can be expressed not only as a matrix data structure suitable for DCNN training but also as a graph data structure. Graph representation can represent entities in strategy games (i.e., player camps, the terrain in the map, in-game resources) as nodes on a graph and relationships between entities (i.e., relationships between camps, distances between resources in the map) as edges of the graph. Compared with matrix data, graph data can not only represent non-Euclidean rules in decision games; it can also consider the adjacency relationship between game players. Graphical representation methods have been applied to rock–paper–scissors games [17], dominance games [18], and strategy games [19]. In the game of Go, Graf and Platzner introduced a common fate graph (CFG) to represent the Go board and extracted features from it to predict moves with the Monte Carlo tree search method [20]. These methods effectively represent non-Euclidean games and enhance the relevance of game rules in strategy games.

Recently, the graph neural networks (GCNs) have attracted more and more attention because of their outstanding processing ability with respect to graph domain data. The GCN mimics the CNN's principle of convolution over structured data and is capable of handling non-Euclidean data that the CNN cannot handle [21]. The GCN has become a crucial graph analysis approach for a wide range of tasks with complex relationships, i.e., medical image analysis [22], location recommendation [23], social network [24], and remote sensing classification [25]. Compared with CNN, because the number of neighbors of the graph data node is different, it is not possible to directly define the sliding window of uniform size, so the GCN is also used to explore the correlation between entities in the image by aggregating the characteristics of the neighbors around the node. Because of the strength of the GCN in these areas, in this paper we proposed to use a GCN to extract dependencies between camps in strategy games and explore the potential impact of camp distribution information on game strategy.

In order to effectively solve the limitations of a standalone deep convolutional neural network (DCNN) in learning game strategy, we propose a hybrid approach to the graph neural network (GCN) to extract the features of the model of game-playing strategies and fuse it into the DCNN. In this paper, this hybrid network will be validated in the game of Go. As shown in Figure 2, first, we encoded the game data into an array data structure and extracted the plane features of the board according to the DCNN. Then, we introduced a method called the common fate graph (CFG) method [26]—which is a graph representation based on the Go rules of sharing liberty for stones of the same color, combining pieces of the same color with straight lines in the same vertex—to represent the Go data and understand the correlation and spatial position characteristics between Go stones based on GCNs. Unlike traditional methods based on DCNNs, our proposed model can better predict moves using the relationships between stones and their positions on the board. The main contributions of this paper are as follows:

1. We propose a novel hybrid deep neural network learning framework that integrates the DCNN and the GCN to extract planar, spatial, and relational features from games;
2. Our use of the graph data structure to represent the game state emphasizes the importance of the relationship between collectives and spatial position in game strategy;
3. We develop an integrated method for the fusion of graph data features and image data features and solve the problem of the incompatibility of feature data in DCNNs and GCNs;
4. We apply a hybrid model to the classic strategy game "Game of Go" and achieve better performance than traditional methods.

The rest of the paper is organized as follows: works related to graph-based learning and methods of hybrid models will be discussed in Section 2; Section 3 introduces our proposed framework in detail; Section 4 discusses the experiment of a hybrid model that we proposed to apply to the Go game; and finally, Section 5 summarizes the article and discusses the direction of future work.

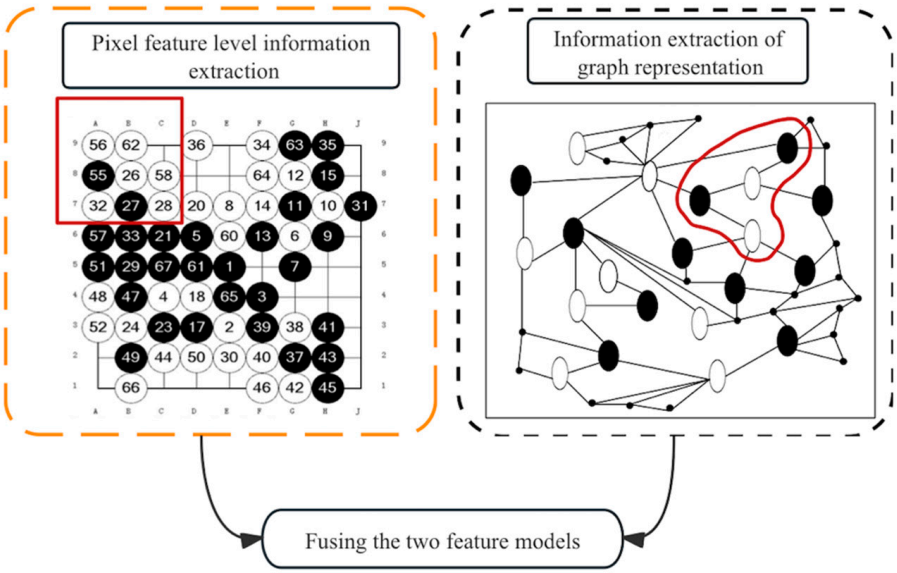

Figure 2. Overview of the proposed methodology.

2. Related Works

Machine learning is considered a data-driven task and has proven effective in many fields. However, in recent years, with the study of graph data, researchers have found many features and contents in graph data that traditional machine learning methods cannot explore. Therefore, graph-based learning has attracted much attention and has been applied to a series of tasks such as graph classification, aggregation, regression, link prediction, etc. For example, in the time and spectral domain, Defferrard et al. [27] proposed the representation of CNN in the context of spectral graph theory. They designed a fast-locating convolution filter on the graph for learning local and fixed features on the graph. Moreover, based on the original network, Kipf and Welling [28] proposed a variant based on the local first-order approximation of spectral graph convolution to improve the selection convolution architecture, which significantly improves the GCN with respect to the node classification task. A critical factor in the CNN's success is training a deep network model. However, the multi-layer GCN causes the extraction to disappear, leads to the excessive smoothing of fixed points, and converges vertex eigenvalues to consistent values, resulting in the current GCN architecture needing to be improved. Moreover, the receptive field of the shallow GCN is limited. To solve these limitations, by using the concept of the CNN, Li et al. [29] applied residual and dense connections and dilated convolutions to the GCN architecture and successfully built a 56-layer GCN with significantly improved performance in semantic segmentation tasks. This research is of great help to the expansion of the GCN in many fields.

Furthermore, graph learning models are important in space and spatial domains. Gao et al. [30] proposed that a learnable graph convolution layer (LGCL) can automatically select a fixed number of adjacent nodes for each feature based on value ranking to enable convolution operations. This method has achieved good results in the subgraph classification of protein structure, which focuses on spatial structure. In addition, Mosella-Montoro and Ruiz-Hidalgo [31] proposed the SkinningNet framework, which utilizes multi-aggregator graph convolution to extract features in an end-to-end learnable fashion to help generalize invisible topologies. In games, the GCN model also performs better in predicting game outcomes than traditional deep learning models. Li et al. [32] represented TUBSTAP game states as graphical data and utilized the GCN as a value network to predict game outcomes through supervised learning, improving the accuracy of results prediction compared to the CNN. With the popularity of multi-agent games, Liu et al. [33] proposed G2AneT, a game mechanism based on the graph attention network for games with complex relationships, which shows the importance of complex relationships between game agents and achieves

better performance in multi-agent games. Furthermore, Lee et al. [34] designed a graph neural network (GNN)-based framework for decentralized strategy problems in multi-agent defense games to mimic expert strategy, with results that outperform other algorithms and can be generalized to large-scale games. From the perspective of semi-supervision, Bisberg and Ferrara [35] proposed a semi-supervised GCN prediction model, GCN-WP, which included more than 30 functions related to the game and achieved good performance in victory prediction. These related works demonstrate the feasibility of graph convolutional networks in games.

With the development of graph learning technology, researchers can analyze the advantages and disadvantages of graph learning in various fields. As a result, the hybrid model approach of fusing deep learning models and graph learning models has become more popular. For example, in hyperspectral image classification, the GCN can use the correlation between adjacent land cover to convolve irregular regions, but it cannot capture features at the pixel level. Therefore, Liu et al. [36] propose a hybrid network called the CNN-enhanced GCN (CEGCN), which generates complementary spectra at the pixel and superpixel levels, respectively, and which performed well on three datasets. Not only that, Wang et al. [37] proposed a method based on a dual-coupled CNN–GCN structure to connect HS and LiDAR data processed by a CNN to construct a graph structure and extract its structure information by sharing some layers of graph structure through a GCN. This method greatly improves performance compared to the traditional remote sensing image classification model. Due to the nature of the CNN model, the spatial information of the data is challenging to capture. Meng et al. [38] developed a multi-stage aggregation network that connected the CNN with the attention refinement module (AR) and the GCN model to enhance the spatial information transmission of ultrasonic medical image data and significantly improve the accuracy of medical image segmentation tasks.

Based on this framework, Duan et al. [39] proposed a GACNN model composed of a CNN, a GCN, and an attention mechanism to extract the global and spatial features of fundus images and improve the network generalization in the classification task by using a semi-supervised method, which significantly improved the model's classification performance in fundus lesion images. Due to the complex parameter tuning of fusion models, Wang et al. [40] proposed an end-to-end multiscale convolutional neural network–dynamic graph convolutional network (AMCNN-DGCN) model that can capture highly discriminative features, and its model can be used to detect the brain waves of driving fatigue with 95.65% accuracy. The GCN can also be integrated with other deep learning models. Li and Yu [29] introduced diffused convolutional recurrent neural networks (DCRNNs), which could capture spatial dependencies using graphically bidirectional random walks and time dependencies using encoder–decoder architectures with planned sampling for traffic prediction. This model extended the time-dependent extension of the graph learning model. Hybrid models based on graph learning have been applied in many fields, such as biology [41], pharmacy [42], network interconnection [43], etc.

Therefore, based on the research of the hybrid model, we propose a framework for the fusion of the DCNN and the GCN, utilizing the advantages of the DCNN in plane feature extraction and the GCN in processing spatial data and entity correlation in games. This paper will take the game of Go as an example for which to build a graph model of games and validate the enhancement effect of GCN on traditional deep learning methods on game strategy.

3. Proposed Methodology

This paper proposes a new hybrid network to integrate the DCNN and the GCN and applies it in the game of Go to improve move prediction over the traditional DCNN. Figure 3 shows the architecture of the proposed hybrid network. It contains plane features extraction based on the DCNN, extraction of graph features in CFG based on the GCN, and features fusion. In the DCNN part, the method of one-hot encoding was used to map the stones on the board to a vector that has the length of the board multiplied by the board's

width. Then, the vectors were input into the DCNN to extract plane features. As part of the graph features' extraction, we utilize a common fate graph model to represent the Go data and input them to R-GCN (relational graph convolutional networks) to extract graph features which include the position feature of stones and the edge feature of the relationships between stones in Go data. Finally, the features extracted via the two models are fused and transferred to a fully connected layer as the classifier in the model to make the move prediction.

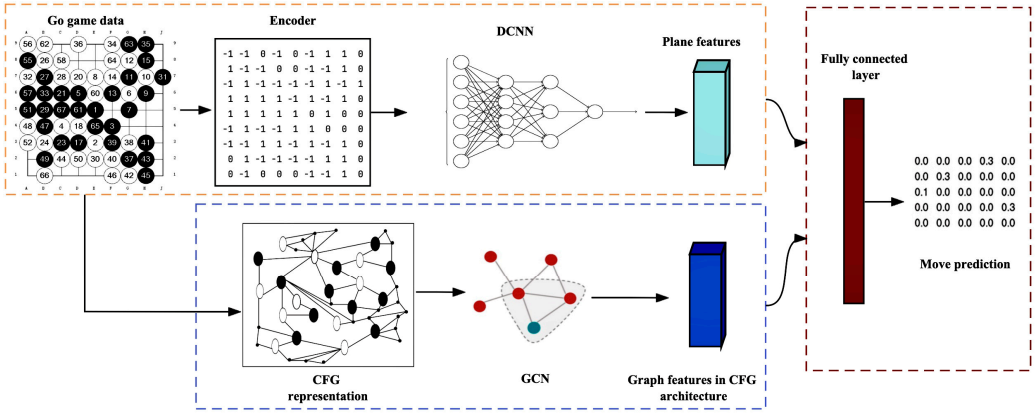

Figure 3. The architecture of the proposed method.

3.1. DCNN in Feature Extraction

Unprocessed Go data are represented by the Smart Game format (SGF), which consists of the metadata of the game and the moves played. The framework specifies metadata by using uppercase letters, encoding the attribute and the corresponding value in square brackets. For example, the black in the third column of the seventh row is represented by B[gc], where B is the stone color, and the coordinates of the rows and columns are indexed alphabetically. This cannot be directly used for network training, so we use the one-hot encoder method to map the positions of Go stones to vectors that include the length and width of the board, as seen in Algorithm 1.

Algorithm 1 Encode Go data to vector

Input: game state of current play which includes the position tuple of move in the Go board [tuple(x_1, y_1), $\cdots$, $tuple(x_n, y_n)$], and board size.
Output: board tensor with 11 plane features v.
Initialize board tensor with the size of 19×19 zero vector v.
if tuple(x_{i+1}, y_{i+1}) is black stone **do**
  Add feature: $v[8] = 1$
else $v[9] = 1$
for r = 1 to range (1 to the length of the board)
  for c = 1 to range (1 to the width of the board)
    $p = position\ tuple(row = r + 1,\ col = c + 1)$
    Find tuple(x_i, y_i) whether has a stone group
    if tuple(x_i, y_i) do not in a stone group **do**
      if tuple(x_i, y_i) is not in ko situation (illegal move) **do**
        $v[10][r][c] = 1$
    else liberty_plane = $min(4,\ stone's\ liberty\ in\ group) - 1$
      if color of Go group == white **do**
        liberty_plane += 4
      $v[liberty\_plane][r][c] = 1$

Moreover, to make this domain effective, we design some additional features in the vector. Due to zero padding in the network, the edge feature of stones is ignored. Therefore,

the first plane feature is describing the edge stones; the second four feature planes describe black stones with one, two, three, or four liberties; and the third four feature planes describe the white stones, which represent the same liberties as the black. Moreover, the remaining feature planes are the color of the stone's turns and are reserved for indicating a ko situation. This encoder with 11 planes led to notable improvements in model performance.

In the DCNN part, we built neural networks consisting of eight convolutional layers, all with ReLU activations, as shown in Figure 4. The pooling layers are not added between the convolutional layers because in the second half of the game, the board is gradually filled with stones, and each position of the stones affects the overall plane features of the board. In order to prevent the high layer output from becoming exceedingly small, we introduced the zero pad before each convolutional layer, and the value of strides is set to 2 to ensure that the features of the stones at the edge of the board can also be extracted. Therefore, the formula for calculating the size of the feature graph generated after convolution is as follows:

$$w' = \frac{(x + 2p – k)}{s},$$

where the input matrix size is x, and k is the convolution kernel size. s denotes the step length, and the number of zero-padding layers is p.

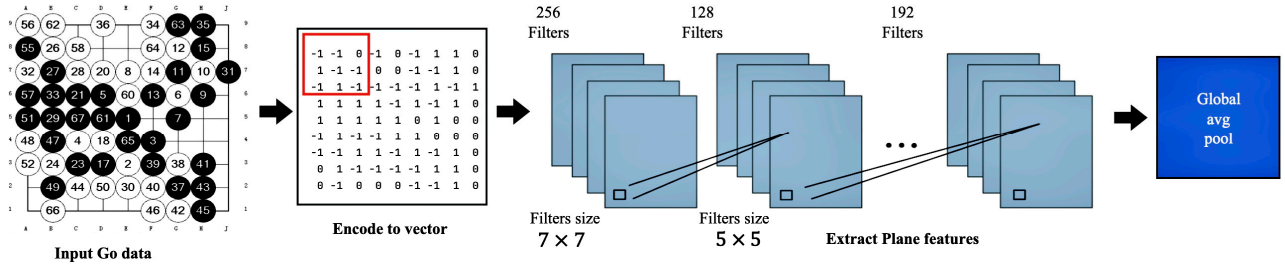

Figure 4. Architecture of the DCNN part.

Different from traditional frameworks, we do not use a fully connected layer, but a global averaging pooling layer instead. This retains the previous convolution layers, extracts spatial information, can be used for fusion with subsequent graph features. The equation of global averaging pooling is expressed as follows:

$$V_f = \frac{\sum_{h=1}^{H} \sum_{w=1}^{W} m_{h,w,f}}{H \times W},$$

where m is the feature map from input Go data after convolutional layers, h and w are the length and width of the data vector, respectively, and f is the number of feature channels. Through the global average pooling layer, the features tensor of input Go data with the size of $h \times w \times f$ can be transferred to a feature vector with the size of f-dim. Due to this method, hybrid networks can reduce a large number of weight parameters and maintain detailed information in Go data.

3.2. Construct the Graph Data Structure

The traditional plane features cannot represent the connection relationship between stones, so we need to represent Go as a graph structure for learning these features. Because of the immutability of the Go board with respect to rotation and mirroring, the position of the board can be expressed as a full graph representation (FGR) with the structure of a 19 × 19 grid in the graph. However, this representation occupies ample memory space, and the expression effect is similar to the planar network structure used to train the DCNN. To improve this approach, we represent the Go board in a common fate graph (CFG), which represents adjacent stones with the same liberty together as a node, and the connection of these stones clusters as one edge based on the rules of the Go "group". This representation

comes from the fact that the straight lines of the same color stones on the board must either survive together or lose their liberty and all be captured, so these stones have a common fate on the board. The method is represented as follows:

$$(v_1, v_2, \cdots v_n) \rightarrow V \ (e_1 \in v_1, e_2 \in v_2, \cdots e_n \in v_n) \rightarrow E \in V,$$

where v is a stone in the stone group sharing a common liberty, and e is the connection of the same stones to the other group in their own group. The CFG not only retains the characteristics of a single stone but also reduces the space of the graph. Moreover, this representation also perfectly defines the territory rules with respect to winning at Go.

Not only was the Go data built into the CFG structure; we also manually added some features to the CFG based on the rules of Go. There are six node features and one edge feature to help the model to better learn the graph structure, as shown in Table 1. These features are divided into binary values and into different vectors. For example, regarding the color of stone, black, white, and empty are split into three different vectors and are represented as binary values. After extracting node and edge features, this approach can reduce the contradiction between GCN and plane features.

Table 1. The features of the CFG.

Type	Features	Num of Vectors	Details
Node	Black, white, empty	3	The color of the stones-occupied intersections
Node	Per liberty in stone group	4	The liberty in the stone group divided by the total number of stones
Node	Legal move	1	Whether the move of stone is legal or not
Node	Turns of move	5	The number of moves in the group
Node	Rewarding move	1	Whether a move results in the capture of an opponent's stone
Node	Number of captures	8	How many of the opponent's stones have been captured
Edge	Relationship of groups	3	Whether the connected stone group is an opponent or empty point, and if it is an opponent, whether it is greater than it

3.3. Branch Model Based on Graph Learning

The Go data represented in the CFG are non-Euclidean data, which means they have no translation invariance, and traditional deep learning models struggle to learn their features. In order to extract the connection relationship between Go stones, we utilize the relational graph neural network (R-GCN), which is a graph neural network, to train the Go data represented by CFG. The common fate is a graph $g = (v, e)$. v and e represent the nodes (stones groups) and edges (stones connections in a group), respectively. In previous studies, Ralaivola et al. [44] extracted game strategy features from CFG models based on enumeration paths and used support vector machines to help classify expert moves. In this paper, we construct an R-GCN model with two graph convolutional layers to extract the graph features. N represents the number of CFG stones groups, and matrix A represents its adjacency matrix, which is also the relationship between the groups. X is the $N \times D$-dimensional matrix composed of the features of these groups, where D is the degree matrix of the adjacency matrix, $D_{ii} = \sum j A_{ij}$. X and A are the inputs in our model based on

the GCN section. Therefore, as a neural network layer, the GCN propagates between layers in the following ways [28]:

$$f(H^L, A) = \sigma(D^{-\frac{1}{2}}\widetilde{A}D^{-\frac{1}{2}}H^L W^L),$$

where $\widetilde{A}$ is $A + 1$, and the core of spectral convolution is the use of symmetric normalization Laplacian matrices which are represented as $D^{-\frac{1}{2}}\widetilde{A}D^{-\frac{1}{2}}$.

The GCN updates the hidden state of a node by weighting its neighbor features and its own features [28]. The hidden state of each node I passing through the graph convolution layer of a traditional GCN can be expressed as the following formula:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in N_i} \frac{1}{c_i} W^{(l)} h_j^{(l)}\right),$$

where $W^{(l)}$ is the shared weight of all edges in the l layer, and h_j is the hidden state of the neighbor of node i. $\frac{1}{c_i}$ is a regularization constant, and σ is the activation function. However, the relationship between different stones nodes is weighted, which is calculated based on the comparison of the number of Go stones contained between nodes of different colors.

To aggregate edge information in a common fate graph, the R-GCN makes an improvement on the original basis, so that the edges of different relations use different weights, only the edges of the same relation type r use the same mapping weight $W_r^{(l)}$. Therefore, the formula of hidden state of node is represented as follows [45]:

$$h_i^{(l+1)} = \sigma\left(W_0^{(l)} h_i^{(l)} + \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)}\right),$$

where $W_0^{(l)} h_i^{(l)}$ represents the updating of the information of the node i, and N_i^r represents the set of neighbor nodes of node i under the relation of $r \in R$. Through the convolution operation of the R-GCN model, the information between adjacent nodes is transferred, and the connection relationship between nodes is also brought into focus. This method allows us to effectively extract the relationship between the stones and the spatial relationship between the territories in the board. Then, we add a global maximum pooling layer to the last layer to prepare for plane feature aggregation. Algorithm 2 shows the proposed method for move prediction.

Algorithm 2 The proposed method for move prediction

Input: Plane vector of Go game data D; batch size s; number of epoch in model training E; learning rate η; number of graph neural networks' layers l.
Output: The accuracy of move prediction.
1: Train DCNN model by training data of 11 plane features vector D.
2: Fine-tune DCNN model by validation dataset.
3: Generate $f\text{-}dim$ plane feature map through global average pool.
4: Construct the CFG, G= $[x, edge_{index}, pos, edge_{attr}, y]$ by manually adding node features and edge features based on the Go game architecture.
5: **for** e = 1 to E **do**
6: Training on R-GCN model in l layers.
7: Calculate loss based on cross-entropy loss function and update weight W through SGD.
8: **end for**
9: Calculate the final output through fusion work and conduct move prediction

3.4. Hybrid Model with Feature Fusion

The hybrid network combines the advantages of graph convolution networks (GCNs) and deep convolutional neural networks (DCNNs). The DCNN model extracted the plane features of the Go data, and the GCN learned the neglected features of the game by modeling the dependencies between the game elements and complex territory information. In order to integrate these two features, we add a feature fusion operation to fuse the features and transfer them to the final fully connected layer for the prediction of moves. The process of fusion with two feature descriptors can be described by the following:

$$F = C^w \cdot (I \oplus \mu G),$$

Where F is the fusion feature, and C^w is a weight parameter in a fully connected layer. I and G are the features that are extracted in the DCNN and R-GCN, respectively. The symbol $\oplus$ denotes an operator in feature fusion for concatenation. The symbol μ represents the fusion ratio to balance the features from plane and graph, and we set it to 0.7 after the experiment.

Finally, we utilize the softmax layer to predict the probability of the moves in each board position. To compute the softmax function for a vector $x = (x_1, x_2 \cdots x_l)$ from the full connection layer, the exponential function is applied to each component to compute e_i^x; then, each of these values should be normalized:

$$softmax(x_i) = \frac{e_i^x}{\sum_{j=1}^{l} e_j^x},$$

where each of these values is the sum of all values to be normalized. By definition, the components of the softmax function are non-negative and add up to 1, meaning the softmax returns probabilities.

Due to the game of Go having a maximum of 361 possible moves, the cross-entropy loss function is leveraged to optimize the parameters to train the hybrid model. It can be expressed as follows:

$$L = \frac{1}{N}\sum_i L_i = -\frac{1}{N}\sum_i \sum_{c=1}^{M} y_{ic} log p_{ic},$$

where M denotes the number of moves in the board. The symbol p_{ic} is the probability of moves. If the true class of sample i equals c, y_{ic} takes 1; otherwise, it takes 0.

4. Experiments

In this section, we describe the details of a hybrid model integrated with the DCNN and the GCN and conduct experiments on the KGS Go dataset for predicting moves. The feasibility of our hybrid network is reported by comparing it with the DCNN model in the original research. The accuracy of the expert's moves is used as a criterion by which to evaluate the model's performance. We also conducted a series of experiments on hybrid networks, including on the influence of different features of the model, and we evaluated the maximum benefit of graph features on the model's move prediction performance after several Go players' moves had been undertaken, as well as the reasonable confidence interval of the predicted value.

4.1. Experiment Dataset

In this experiment, we used the Go dataset from the KGS Go server on the webpage http://u-go.net/gamerecords/ (accessed on 28 July 2023), which collected the Go game records from games played by high-ranking players from 2001 to 2019. It contains about 179,689 expert games (about 4 million move locations) played on boards with a size of 19×19 in which at least one of the players was 7 dan or above, or in which both players were 6 dan. Based on the previous experiment [46], we divided the dataset into the training,

validation, and test sets, accounting for 90%, 2%, and 8% of the dataset, respectively. Validation sets are used to regulate hyperparameters by monitoring learning and finding the parameters corresponding to the best-performing model. In the dataset, the entire game of Go is shuffled, not the state of individual game moves, because the moves within a single game each depend on one another.

4.2. Details of Implementation

The hybrid networks were constructed in the PyTorch framework and Python-3.9 with Intel Core i7 CPU and NVIDIA GTX 3050Ti GPU. To further improve the performance of the move prediction model, we used the stochastic gradient descent (SGD) as an optimizer. The SGD optimizer was set with a learning rate of 0.001, a 0.1% decay rate, and 90% momentum. The hybrid model is composed of the DCNN and GCN models, and traisn with a batch size of 128 for 10 epochs. Table 2 shows the details of the number of layers and the size and number of filters in each model.

Table 2. Configuration of hybrid mode.

Model	Layer	Num of Filters	Size of Filter
DCNN	3 layers	256	7×7
	5 layers	128	5×5
	2 layers	192	5×5
R-GCN	1 layer	256	

4.3. Experiment Results

We performed several experiments during this analysis to ensure that the results were stable and not stochastic. The basic DCNN model is used as a baseline by which to compare the fusion model with the GCN model, and the accuracy of the move prediction is the evaluation criterion by which we evaluate the reliability of the novelty model. Due to the dataset being disrupted, and the Go strength (rank) of the players being irregular with respect to one another in the Go records, it was difficult for the rank of the games in the training set and test set to be the same, so we conducted average processing. Rank represents the Go level of the players; the higher the rank, the more potent the player's strength in Go. The mean and standard deviation of all experiments were calculated to evaluate whether the accuracy was within a reasonable range. We also calculate the standard error (stderr) of the experiment's accuracy via random sampling, which makes the experiment more persuasive.

4.3.1. Comparison of Different Models

To validate the performance improvement of the hybrid networks over the traditional DCNN, we conducted experiments in which the CNN–GCN model and the traditional DCNN had the same number of convolution layers and other parameters as the DCNN model of the hybrid model. We measured the move prediction accuracy in the training set and test set for 10 epochs, and the loss of each epoch is shown in Figure 5. The mean and standard deviation of the experiment on move prediction are used to evaluate the rigor of the experiment, as shown in Figure 6. Table 3 shows the difference in accuracy between the two models according to Chinese rules. Through training, our proposed model can reach a maximum accuracy of nearly 50% in predicting moves on the KGS GO dataset. It is worth mentioning that under the same test set verification, the performance of the hybrid model is improved compared with the traditional DCNN model, and the accuracy is increased by approximately 2%. To our knowledge, the previously known work on the KGS dataset adopted a similar algorithm using a smaller 8-layer DCNN model and reached approximately 45% accuracy with respect to move prediction [15]. Although the dataset applied by our model has reached its most recent version compared to previous ones, the

predicted results are still in the comparable range. Compared with the prediction result of the traditional DCNN model, our hybrid model's accuracy exceeds 4.43%.

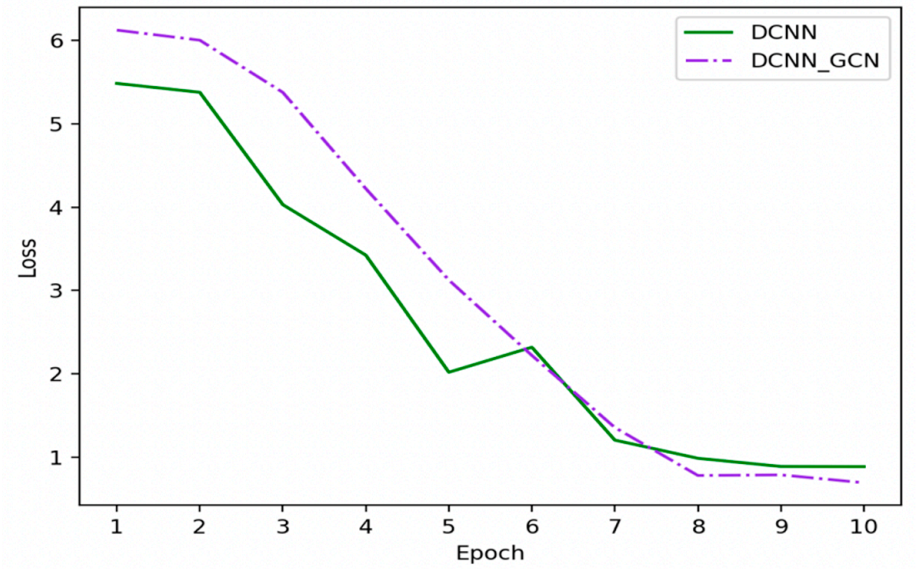

Figure 5. Loss of each epoch.

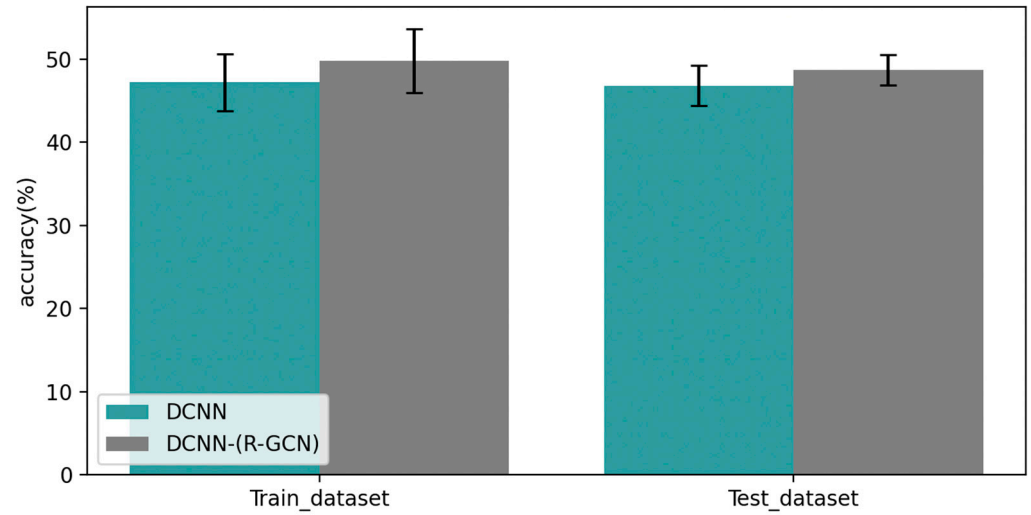

Figure 6. Mean and standard deviation from randomness runs.

Table 3. Move prediction in different models.

Model	Test Dataset			Train Dataset		
	Accuracy	Rank	Stderr	Accuracy	Rank	Stderr
DCNN	46.8%	6.87	±1.2%	47.2%	6.35	±1.7%
DCNN-(R-GCN)	48.7%	6.87	±0.9%	49.8%	6.35	±1.9%

4.3.2. Performance of Feature Extraction

We also conducted several experiments on the hybrid model to validate the correlation effect between Go stones on the move strategy. As shown in Table 4, it was apparent from the prediction results that the CFG data for building relationship weights between groups of stones showed better performance after training. Moreover, the construction of plane features in vector data also affected the learning of the hybrid model. Based on the principle

of R-GCN, feature extraction on the common fate graph only affects the weight division between different relations, and other parameters are no different from the GCN model. This shows that the DCNN model has a weakness in analyzing group relationships in game strategy, but the R-GCN can enhance the performance of the ordinary DCNN model in this direction. Due to the shuffling of the dataset and the randomness of the experiment, we also calculated each model's mean and standard deviation after multiple experiments, as shown in Figure 7.

Table 4. Accuracy of models with different features.

Model	Features	Accuracy	Stderr
DCNN	None	43.1%	±3.0%
DCNN	11	46.8%	±1.2%
DCNN–GCN	11 plane features No Edge feature	47.9%	±1.5%
DCNN–(R-GCN)	11 plane features Have Edge feature	48.7%	±1.9%

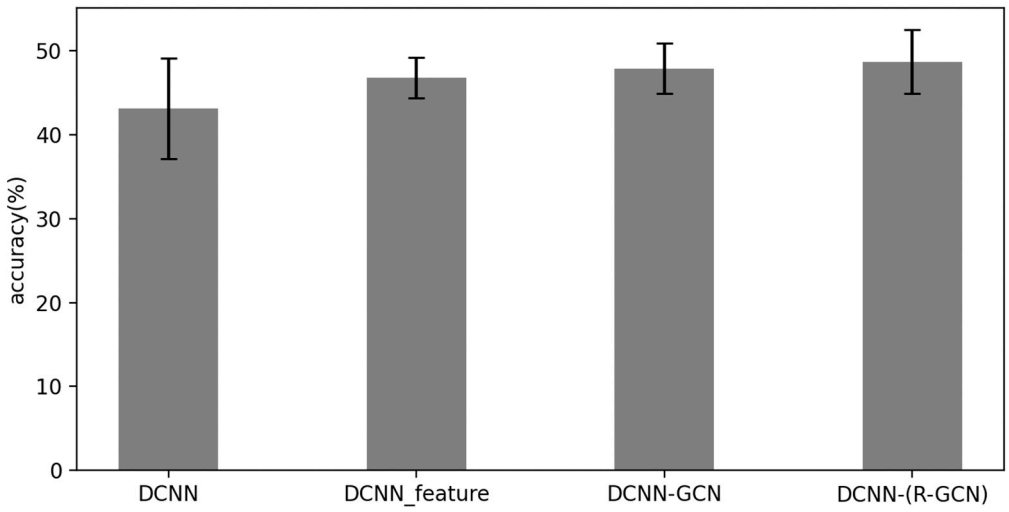

Figure 7. Mean and standard deviation of models with different features.

4.3.3. Performance of Hybrid Graph Convolution Neural Networks

In addition to the accuracy of the moves as a criterion for model evaluation, we also divided the game into 10 stages, each containing 36 expert moves (because the maximum number of moves on the board is only 361) to test the performance of the hybrid model in game playing. In Figure 8, we can see that since there are very few stones on the board at the beginning, it is difficult for the model to extract useful information, and the relationship between the stones in the common fate graph will lose its function. Until the halfway point of the game is reached, with the gradual increase in the number of Go stones on the board, the GCN's extraction of the relationship features in the graph begin to affect the model's training and improve the hybrid model's performance. This shows that the relationship between the stones has a significant impact on the evaluation of game strategy. Thus, the more the game progresses, the better our proposed model performs in move prediction.

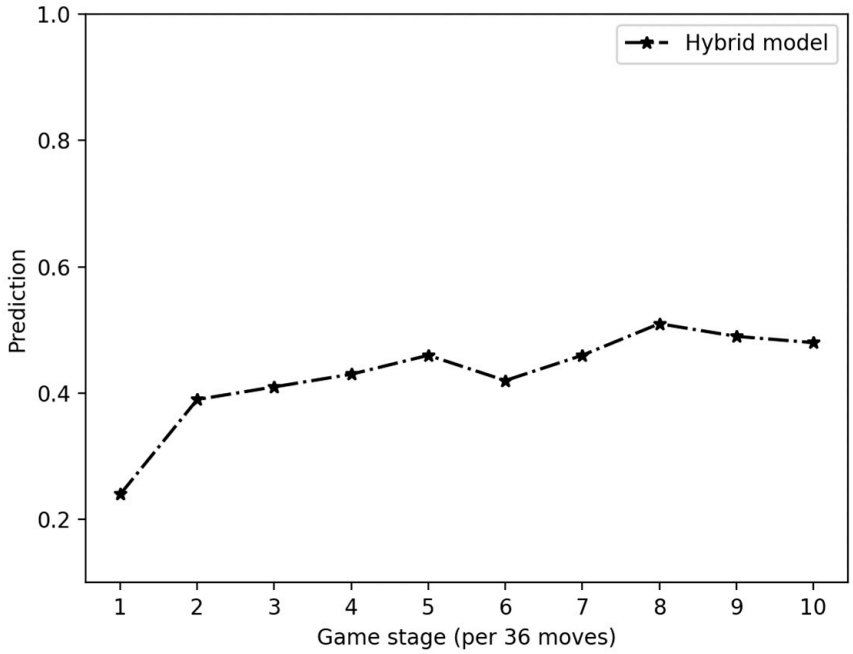

Figure 8. Predictions of expert moves at each stage.

Next, the confident prediction within the number of n guesses is also a worthy criterion by which to evaluate the hybrid network. If n is small, this means that the network can predict the outcome of the expert's moves within a small confidence interval. As shown in Figure 9, predictions of expert moves become more accurate with more guesses and converge after about 25 guesses. This shows that the probability of predicting correct expert moves is in the top 25 for the hybrid model. However, in an actual game, the referee will not give 25 more chances for the network to guess on the Go board; therefore, compared with actual 7 Dan experts, there is still a notable gap in the model we proposed. As an auxiliary tool for building game strategies, hybrid model predictions after multiple attempts can help game programs reduce the search space effectively.

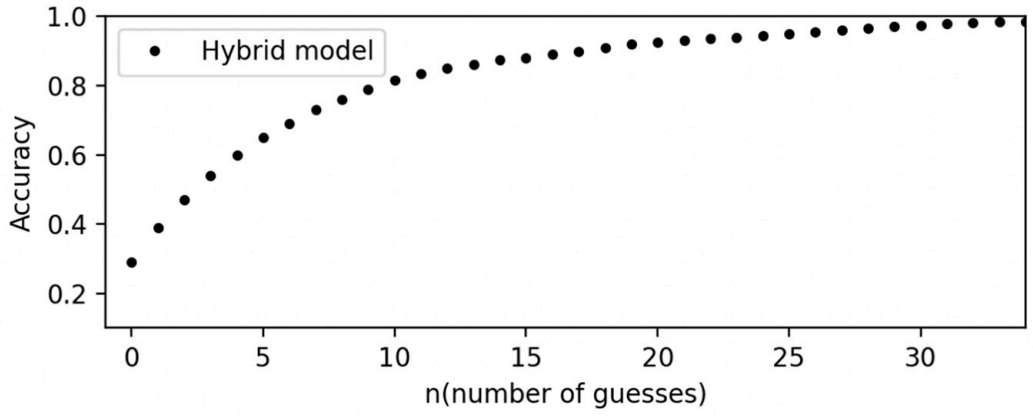

Figure 9. Top-n predictions of expert moves.

4.4. Theoretical Applications in Other Strategy Games

Based on the results of the above evaluations of experiments applied in cases of the game of Go, the hybrid model that we proposed shows excellent performance and potential for other strategy games. As a milestone for strategy games, most strategy games are based on the foundations of Go, so the game of Go contains the characteristics that most strategy games possess. The performance of the hybrid model in the game of Go indirectly illustrates

the feasibility of the model in other strategy games. For example, in the board game, "space four chess", whose pieces have spatial characteristics, it is necessary to construct graph data in order to represent the relationship and position of chess pieces in 3D space, as shown in Figure 1. In multi-agent strategy games, such as "League of Legends" and "Dota 2", these games involve teams of heroes battling each other, which, as in Go, requires the hybrid model to consider not only the choices of individual agents but also the strategies of the entire group. Furthermore, in strategy video games such as "Starcraft" and "Age of Empires", the hybrid framework can leverage the CNN to deal with the large map in the game, and the GCN structure can determine the impact of the terrain in the map. Due to the nature of the graph architecture, the GCN can also analyze the relationship between the group and the game unit to make strategic decisions. The properties of these different strategies are all contained in the game of Go, and in the experiment, it can be seen that they are well solved by the hybrid model.

5. Conclusions and Future Work

In this paper, we designed a hybrid network integrating the GCN and the DCNN and applied it to a classic strategy game—the game of Go—to validate our work. This hybrid model not only utilizes the advantages of deep convolutional neural networks in extracting plane features but also introduces graph neural networks to process graph structure data constructed according to the game's rules and learns the correlation and dependency between the entities in the game. The game of Go, as a complex strategy game, is just a test for our proposed framework; in simple spatial strategy games, the fusion framework will show better performance with respect to strategic decision-making. Based on this experiment, the following are some of the advantages of the hybrid model in developing strategies in games:

1. Enhance the identification of strategic situations: The model can identify common formations that indicate strong and weak positions, enhancing the impact of a team's overall strength on the game as a whole rather than that of a single individual.
2. Assess the game on multiple levels: The model can consider regional characteristics based on the graph structure, explicitly capturing the connections between game nodes (relationships, rewards, resources), their territorial impact, and potential correlations.
3. Adapt to different stages of the game: Strategy games have different stages, such as the opening, mid-game, and end game. The versatility of the hybrid approach in capturing different aspects of the game allows it to adapt to the strategic needs of these different phases. For example, the GCN can learn the long-term dependencies of games, while the CNN is good at learning tactical patterns.

Therefore, the framework we propose solves the problem of traditional deep learning models ignoring the relational features and spatial location features of entities when learning game strategies.

While GCN-assisted hybrid models have shown significant improvements in game strategy-making compared to traditional methods, some limitations still need to be addressed in future work. First, at the level of feature fusion, the method we used is relatively simple and does not consider the particularity of some strategy games in terms of rules. Therefore, more general fusion methods will be explored in the future. Second, in this article, our approach to building graph-structured data for games using the CFG is computationally complex and requires the manual features of nodes and edges. To alleviate this problem, a potential solution is to use the superpixel function to build the game's graph data in a follow-up experiment, which is suitable for future expansion to other strategy games. Third, at the level of graph-based learning, many game states are not fixed from time to time, so our subsequent work will introduce a time series approach to the hybrid model to deal with strategy games that need to be structured as dynamic graph data. Moreover, because of the principle of GCNs, the GCN's ability to perform in some games is limited, so in future work, we will explore more graph-based learning models. In

summary, potential future works will involve the addition of superpixel and other methods to make the hybrid framework more universal and applicable to different strategy games.

Author Contributions: Conceptualization, K.P.S., H.X. and L.-M.A.; resources, K.P.S. and L.-M.A.; writing—original draft preparation, H.X. and K.P.S.; writing—review and editing, K.P.S., H.X. and L.-M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The research dataset is KGS Go. Link: https://u-go.net/gamerecords/.

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

References

1. 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]
2. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv* **2017**, arXiv:1712.01815.
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. Neven, F.; Schwentick, T.; Vianu, V. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.* **2004**, *5*, 403–435. [CrossRef]
5. Browne, C.; Powley, E.J.; Whitehouse, D.; Lucas, S.M.; Cowling, P.I.; Rohlfshagen, P.; Tavener, S.; Liebana, D.P.; Samothrakis, S.; Colton, S. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. Comput. Intell. AI Games* **2012**, *4*, 1–43. [CrossRef]
6. Furukawa, M.; Abe, M.; Watanabe, T. A Study on Utility Based Game AI Considering Long-Term Goal Achievement. *J. Soc. Art Sci.* **2021**, *20*, 139–148. [CrossRef]
7. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Driessche, G.V.; 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]
8. Hazra, T.; Anjaria, K. Applications of game theory in deep learning: A survey. *Multimed. Tools Appl.* **2022**, *81*, 8963–8994. [CrossRef]
9. Mahajan, C. Reinforcement Learning Game Training: A Brief Intuitive. *MatSciRN Other Electron.* **2020**. [CrossRef]
10. Yasruddin, M.L.; Hakim Ismail, M.A.; Husin, Z.; Tan, W.K. Feasibility Study of Fish Disease Detection using Computer Vision and Deep Convolutional Neural Network (DCNN) Algorithm. In Proceedings of the 2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA), Selangor, Malaysia, 12–12 May 2022; pp. 272–276.
11. Hou, J.; Gao, T. Explainable DCNN based chest X-ray image analysis and classification for COVID-19 pneumonia detection. *Sci. Rep.* **2021**, *11*, 16071. [CrossRef]
12. Yang, J.H.; Choi, W.Y.; Lee, S.; Chung, C.C. Autonomous Lane Keeping Control System Based on Road Lane Model Using Deep Convolutional Neural Networks. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 3393–3398.
13. Wang, B.; Ma, R.; Kuang, J.; Zhang, Y. How Decisions Are Made in Brains: Unpack "Black Box" of CNN With Ms. Pac-Man Video Game. *IEEE Access* **2020**, *8*, 142446–142458. [CrossRef]
14. Sutskever, I.; Nair, V. Mimicking Go Experts with Convolutional Neural Networks. In Proceedings of the International Conference on Artificial Neural Networks, Prague, Czech Republic, 3–6 September 2008.
15. Clark, C.; Storkey, A.J. Training Deep Convolutional Neural Networks to Play Go. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015.
16. Li, Z.; Zhu, C.; Gao, Y.; Wang, Z.; Wang, J. AlphaGo Policy Network: A DCNN Accelerator on FPGA. *IEEE Access* **2020**, *8*, 203039–203047. [CrossRef]
17. Ichsan, M.N.; Armita, N.; Minarno, A.E.; Sumadi, F.D.; Hariyady. Increased Accuracy on Image Classification of Game Rock Paper Scissors using CNN. *J. RESTI (Rekayasa Sist. Dan Teknol. Inf.)* **2022**, *6*, 606–611. [CrossRef]
18. Kamatekar, S.L.; Hiremath, S.M. Domination, Easymove Game Represented in Graph. *Int. J. Math. Arch.* **2018**, *9*, 179–185.
19. Yun, W.J.; Yi, S.; Kim, J. Multi-Agent Deep Reinforcement Learning using Attentive Graph Neural Architectures for Real-Time Strategy Games. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021; pp. 2967–2972.
20. Graf, T.; Platzner, M. Common fate graph patterns in Monte Carlo Tree Search for computer go. In Proceedings of the 2014 IEEE Conference on Computational Intelligence and Games, Dortmund, Germany, 26–29 August 2014; pp. 1–8.
21. Xia, F.; Sun, K.; Yu, S.; Aziz, A.; Wan, L.; Pan, S.; Liu, H. Graph Learning: A Survey. *IEEE Trans. Artif. Intell.* **2021**, *2*, 109–127. [CrossRef]

22. Zhai, Z.; Staring, M.; Zhou, X.; Xie, Q.; Xiao, X.; Bakker, M.E.; Kroft, L.J.; Lelieveldt, B.P.; Boon, G.J.; Klok, F.A.; et al. Linking Convolutional Neural Networks with Graph Convolutional Networks: Application in Pulmonary Artery-Vein Separation. In Proceedings of the Graph Learning in Medical Imaging: First International Workshop, GLMI 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, 17 October 2019.
23. Zhong, T.; Zhang, S.; Zhou, F.; Zhang, K.; Trajcevski, G.; Wu, J. Hybrid graph convolutional networks with multi-head attention for location recommendation. *World Wide Web* **2020**, *23*, 3125–3151. [CrossRef]
24. Wilkens, R.S.; Ognibene, D. MB-Courage@EXIST: GCN Classification for Sexism Identification in Social Networks. In Proceedings of the IberLEF 2021, Málaga, Spain, 21–24 September 2021.
25. Liang, J.; Deng, Y.; Zeng, D. A Deep Neural Network Combined CNN and GCN for Remote Sensing Scene Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 4325–4338. [CrossRef]
26. Graepel, T.; Goutrie, M.; Kruger, M.; Herbrich, R. Learning on Graphs in the Game of Go. In Proceedings of the International Conference on Artificial Neural Networks, Vienna, Austria, 21–25 August 2001.
27. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Proceedings of the Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016.
28. Kipf, T.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2016**, arXiv:1609.02907.
29. Li, Y.; Yu, R.; Shahabi, C.; Liu, Y. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. *arXiv* **2017**, arXiv:1707.01926.
30. Gao, H.; Wang, Z.; Ji, S. Large-Scale Learnable Graph Convolutional Networks. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018.
31. Mosella-Montoro, A.; Ruiz-Hidalgo, J. SkinningNet: Two-Stream Graph Convolutional Neural Network for Skinning Prediction of Synthetic Characters. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 18572–18581.
32. Li, W.; He, H.; Hsueh, C.; Ikeda, K. Graph Convolutional Networks for Turn-Based Strategy Games. In Proceedings of the International Conference on Agents and Artificial Intelligence, Online Streaming, 3–5 February 2022.
33. Liu, Y.; Wang, W.; Hu, Y.; Hao, J.; Chen, X.; Gao, Y. Multi-Agent Game Abstraction via Graph Attention Neural Network. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
34. Lee, E.S.; Zhou, L.; Ribeiro, A.; Kumar, V. Learning Decentralized Strategies for a Perimeter Defense Game with Graph Neural Networks. *arXiv* **2022**, arXiv:2211.01757.
35. Bisberg, A.; Ferrara, E. GCN-WP—Semi-Supervised Graph Convolutional Networks for Win Prediction in Esports. In Proceedings of the 2022 IEEE Conference on Games (CoG), Beijing, China, 21–24 August 2022; pp. 449–456.
36. Liu, Q.; Xiao, L.; Yang, J.; Wei, Z. CNN-Enhanced Graph Convolutional Network with Pixel- and Superpixel-Level Feature Fusion for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2020**, *59*, 8657–8671. [CrossRef]
37. Wang, L.; Wang, X. Dual-Coupled CNN-GCN-Based Classification for Hyperspectral and LiDAR Data. *Sensors* **2022**, *22*, 5735. [CrossRef] [PubMed]
38. Meng, Y.; Wei, M.; Gao, D.; Zhao, Y.; Yang, X.; Huang, X.; Zheng, Y. CNN-GCN Aggregation Enabled Boundary Regression for Biomedical Image Segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Lima, Peru, 4–8 October 2020.
39. Duan, S.; Huang, P.; Chen, M.; Wang, T.; Sun, X.; Chen, M.; Dong, X.; Jiang, Z.; Li, D. Semi-supervised classification of fundus images combined with CNN and GCN. *J. Appl. Clin. Med. Phys.* **2022**, *23*, e13746. [CrossRef]
40. Wang, H.; Xu, L.; Bezerianos, A.; Chen, C.; Zhang, Z. Linking Attention-Based Multiscale CNN With Dynamical GCN for Driving Fatigue Detection. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 2504811. [CrossRef]
41. McDonnell, K.; Abram, F.; Howley, E. Application of a Novel Hybrid CNN-GNN for Peptide Ion Encoding. *J. Proteome Res.* **2022**, *22*, 323–333. [CrossRef]
42. Liang, Y.; Jiang, S.; Gao, M.; Jia, F.; Wu, Z.; Lyu, Z. GLSTM-DTA: Application of Prediction Improvement Model Based on GNN and LSTM. *J. Phys. Conf. Ser.* **2022**, *2219*, 012008. [CrossRef]
43. Li, B.; Zhu, Z. GNN-Based Hierarchical Deep Reinforcement Learning for NFV-Oriented Online Resource Orchestration in Elastic Optical DCIs. *J. Light. Technol.* **2022**, *40*, 935–946. [CrossRef]
44. Ralaivola, L.; Wu, L.; Baldi, P. SVM and pattern-enriched common fate graphs for the game of go. In Proceedings of the The European Symposium on Artificial Neural Networks, Bruges, Belgium, 27–29 April 2005.
45. Schlichtkrull, M.; Kipf, T.; Bloem, P.; Berg, R.V.; Titov, I.; Welling, M. Modeling Relational Data with Graph Convolutional Networks. In Proceedings of the Extended Semantic Web Conference, Portorož, Slovenia, 28 May 28–1 June 2017.
46. Maddison, C.J.; Huang, A.; Sutskever, I.; Silver, D. Move Evaluation in Go Using Deep Convolutional Neural Networks. *arXiv* **2014**, arXiv:1412.6564.