Sudden-death prediction using Deep Convolutional Neural Network in Connect6

Jung-Kuei Yang[†1], Shi-Jim Yen[†2], Yu-Yu Yang[†2]

Abstract: This paper describes using Deep Convolutional Neural Network (DCNN) to predict the positions of sudden-death in Connect6. In sudden-death game, if one side cannot identify sudden-death positions, the other side will win the game at next move. Therefore, the prediction of sudden-death positions is of great significance for pruning the branch degree of the search tree. This study proposes many deep CNN model based on the features of Connect6 and trains it by lots of sudden-death positions established from our previous study. Then the best DCNN model is selected from the experimental results. The experimental results show that the depth of stacking multiple convolutional layers is the key influencing factor of deep CNN to predict sudden-death positions in Connect6. The results of this study can improve the search performance of Kavalan, which is an AI program we design to play Connect6 game.

Keywords: Connect6, sudden-death position, Deep Convolutional Neural Network.

1. Introduction

1.1 Connect6

Since 2005, when Prof. Wu [11] proposed the k-in-a-row game or Connect(m,n,k,p,q), the Connect6 has become a very popular research topic. Two players, Black and White, alternately place two stones of their own colour, Black and White respectively, on empty intersections (or cells) of a Go-like board, except that Black (the first player) places one stone only for the first move. The one who gets six or more stones in a row (horizontally, vertically or diagonally) first wins the game. Most often, Connect6 is played on a 19x19 Go board.

Connect6 is a sudden-death game. In sudden-death game, if one side cannot identify sudden-death positions, the other side will win the game at next move. In view of this, identifying the sudden-death position is a very critical point for developing sudden-death games (such as Connect6). In order to avoid falling into the sudden-death positions at the opening of the game, the program must have a relevant algorithm to judge these states.

Some simple sudden-death positions can be obtained by a search algorithm, but complex sudden-death positions cannot be easily found. At present, these complex sudden-death positions are stored in opening database when dealing with sudden-death positions. Although this can solve most of the sudden-death problems, it has two disadvantages. First, there are lots of sudden-death positions; therefore, we have to spend a lot of storage space. Second, sudden-death positions are to use threat space search (TSS) [11][14] and relevance zone search (RZS) [12][13] to determine whether the board state is a sudden-death position. Therefore, determining whether a board state is a sudden-death depends on the program's search capabilities. In addition to spending a lot of computing resources, there is currently no complete algorithm that can find all the sudden-death positions.

1.2 Sudden-death positions in Connect6

Many sudden-death positions require enormous computing resources to identify the sudden-death positions. Fig. 1 is an example of sudden-death position. Under the position of (a), if White cannot identify the sudden-death position of M4 as shown in (b), White chooses to play the move of M4. Black will win the game after it plays the move of M5 as shown in (c). In the state of Fig. 1 (c), White cannot block Black wins; therefore, (c) is a win-position of Black. If a certain position is called a win-position, it represents the move in that position, no matter how the opponent blocks, the side can finally find the winning move order.

†1 School Of Information Science and Technology, Huizhou University, Huicheng District, Huizhou, Guangdong, China
†2 Dept. Of Computer Science and Information Engineering, National Dong Hwa University, Taiwan

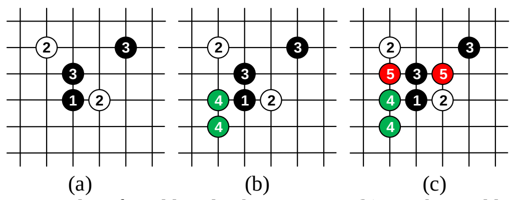

Fig. 1. Example of sudden-death position. (b) is the sudden-death position, and (c) is a win-position. This example diagram shows a section of the board, and the numbers representing the order of moves.

In general, lots of search time could be saved if the search algorithm could predict sudden-death moves from candidate moves at the beginning of the search. These features and the fact that the game has not yet been solved makes Connect6 an interesting test bed for computational intelligence methods. Motivated by the success in computer Go [7], this study investigates how to use deep CNN to represent and learn knowledge in sudden-death positions of Connect6 game.

2. sudden-death prediction with CNN

Convolutional neural networks (CNNs) are one of the best learning algorithms for understanding image content and have shown excellent performance in many fields of application [2][4][6][7][8]. In this section, we first introduce deep convolutional neural network (deep CNN), then our deep CNN architecture, final design input channels and output neurons for deep CNN of Connect6. After that our architecture of deep convolutional layers will be presented.

2.1 deep convolutional neural networks

CNN is a new type of neural network [5] that combines neural network with convolutional layers. In addition to retaining the advantages of automatic learning features of neural network, the computing time can be greatly reduced without sacrificing the accuracy rate, and with the generalization of the graphics processing unit (GPU), better recognition results can be achieved in feasible time.

The event of AlphaGo [7][8] defeating the human grandmaster have a great impact on the board game. The newest techniques that Google used in AlphaGo attract pervasive focuses and interest, which include different deep neural networks. AlphaGo employs deep convolutional neural networks (DCNNs) in deep learning (DL) to identify chess patterns and predict the moves of grandmasters, thereby greatly reducing the search width [7][8].

The topology of CNN is divided into multiple learning stages composed of a set of the convolutional layers, non-linear processing units, and subsampling layers [5]. The powerful learning ability of deep CNN is primarily due to the use of multiple feature extraction stages that can automatically learn features from the data [5]. In the next part, we will introduce the convolutional layers of Connect6 and the overall deep CNN architecture.

2.2 Deep CNN architecture of Connect6

The architecture of a convolutional network typically consists of four types of layers: convolution, pooling, activation, and fully connected [5]. This study stacks multiple convolutional layers to fulfill deep CNN, and it contains d (d ≧ 3) k x k convolutional layers with different number of filters, and k is the size of kernel. The kernel size selected for this study is 3, which is 3x3 convolutional layers. Fig. 2 shows the stacking multiple convolutional layers architecture of this study.

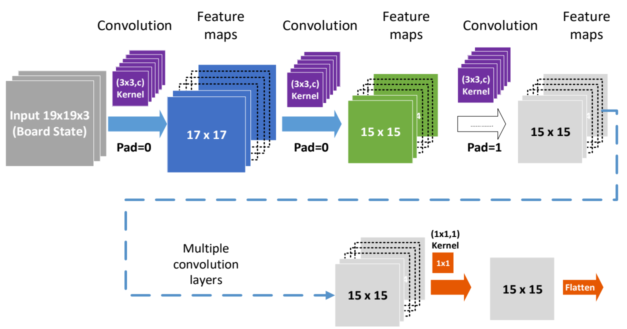

Fig. 2. Stacking multiple convolutional layers in deep CNN architecture. In the figure, the 3x3 kernel with different number of filter (3x3, c) and c is the number of filters. In the DCNN model, the first two convolutional layers have no padding from the border, and the subsequent convolutional layers pad it with zero.

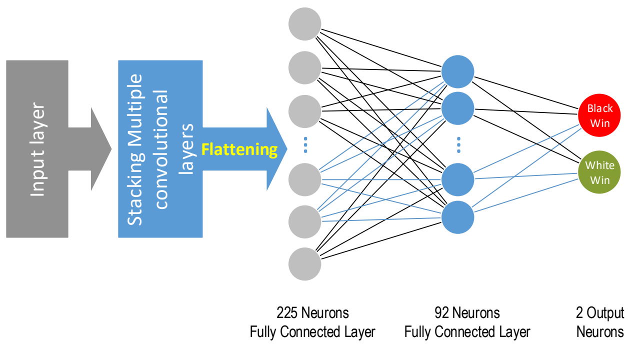
Input layer
Stacking Multiple convolutional layers
Flattening
Black Win
White Win
225 Neurons Fully Connected Layer
92 Neurons Fully Connected Layer
2 Output Neurons

Fig. 3. Fully connected layers in deep CNN architecture. There are three fully connected layers including the output layer. The neurons of the fully connected layer are 225, 92 and 2, respectively.

This study mainly focuses on the prediction of the sudden-death positions at the opening of Connect6 game. Therefore, the first two convolutional layers have no padding from the border. Since the data in the training set, the first-move of Black (only one stone) is placed on the center of the board, and the index number is 180. Therefore, at the opening of the game, the stones in the board will only be concentrated in the central part, so the size of the feature maps after the convolution operation is reduced to 17x17 and 15x15 and will not affect the correctness of feature extraction.

At the end of the convolutional layer, we add a 1x1 kernel with the number of filter is 1 (1x1, 1) for reducing dimensions. Then we flatten the single feature map and transform it into a one-dimensional vector (15x15=225 neurons), followed by the fully connected layers. After stacking multiple convolutional layers, this study uses two fully connected layers and one output layer as shown in Fig. 3.

2.3 Details of each stage in CNN

In this section, we will further explain the details of the implementation of Connect6's deep CNN model.

2.3.1 Input layer

There are three channels (or classes) in the input layer. The structural analogy between bitmap images and board states in games is natural [6]; therefore, this study treats a board state as an image, and it has three channels representing Black, White and Empty respectively. Because each cell in the board has three states, the input of three channels can just be used to represent three different states for each cell of board. They represent the black, white stones occupied in the 19x19 board respectively and the third empty cells.

Each board state is preprocessed to extract three feature channels, which are fed into the CNN. There are 19x19 cells in a board, and the following channels are being extracted:

1. Black: Cell with 1 for black pieces and 0 otherwise.
2. White: Cell with 1 for white pieces and 0 otherwise.
3. Empty: Cell with 1 for empty spaces and 0 otherwise.

In our previous study, the position is stored as a 361-character string in knowledge base, so when the sudden-death position is used as the input of the neural network, it can be converted naturally. For the sudden-death positions in the knowledge base, the first-move of Black is placed on the center of the board, and the index number is 180. We will describe these training data in more detail later.

2.3.2 Convolutional Layers

Convolutional layers use kernels to extract certain features from an input board state or the feature maps. The kernel size is (3x3, c), and c is the number of filters. The stride set to one. We create many feature maps to obtain the features of Board state, and the first two convolutional layers have no padding. Because this study focus on the sudden-death positions of Connect6 opening, the first two convolutional layers has no padding to reduce computing resources.

In addition, the subsequent convolutional layers pad it with zeros on the border, so the size of feature maps will not become smaller after the convolution operation. We use padding from the third convolutional layer, which avoids the problem that the size of the feature maps become too small after the data passes through multiple convolutional layers.

In this study, two kind of the numbers of filters were used with DCNNs to determine their effect on accuracy and learning efficiency of predicting sudden-death positions for Connect6. We will describe these settings in the experimental design.

This study uses rectified linear units (ReLUs) to process convolution outcomes. The ReLUs activation function: f(x) = max(0; x). The vanishing gradients problem [3][10] is one example of unstable behavior that we may encounter when training a deep neural network. When a deep multilayer feed-forward network or a recurrent neural network is unable to propagate useful gradient information from the output end of the model back to the layers near the input end of the model. Therefore, ReLUs are used here as the activation function.

2.3.3 Fully Connected Layers

Pooling is a subsampling technique to reduce the dimensionality of data [7]. In order to preserve the spatial resolution after convolution, this study do not use any form of pooling [2][6]. Therefore, the stack of convolutional layers is followed by flattening operation, and then three fully connected layers. They are two fully connected layers consist of 225 neurons and 92 neurons with ReLUs activations. The second fully connected layer has 92 neurons, representing 92 lines [15] in 4 directions on the 19x19 board. Connect6 is a connection type game, so the connection formed by the four directions on the board is an important information for the game.

2.3.4 Output Layer

The output layer has two neurons corresponding to the win rate of Black or White. For the output values to be interpreted as probabilities they must sum to unity, which is achieved by the softmax transformation. The softmax function takes as input a vector z of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers [9]. After applying softmax, each component will be in the interval (0,1), and the components will add up to 1, so that they can be interpreted as probabilities.

3. Experiments

We consider deep convolutional neural network as the predictor of sudden-death positions, and use *ConvNetSharp* [1] as the implementation and training package for Deep CNN.

ConvNetSharp which is descended from ConvNetJs is a library which enables us to use convolutional neural networks in .NET without the need to call out to other languages or services. ConvNetSharp also supports GPU [1], but you must have CUDA version 10.0 and cuDNN v7.6.4 (September 27, 2019), for CUDA 10.0 installed. cuDNN bin path should be referenced in the PATH environment variable. Because Kavalan is developed under the .NET framework, this study uses ConvNetSharp as the development and training platform for deep CNN.

In this section, we first design our CNN architectures used in the experiments, then introduce training data for CNN training. After that our experimental result will be present.

3.1 Experimental Design

This study compares various configurations of the neural predictors in sudden-death positions described in Section II. The size of kernel is 3x3, and two kind of the number of filters are used 8 and 16: (3x3, 8), (3x3, 16). The number of stacking multiple convolutional layers: 3, 4, 5 (the depth of convolutional layers, and it is not included kernel (1x1, 1)).

TABLE 1 CNN ARCHITECTURES USED IN THE EXPERIMENTS.

Deep CNN Model	Architectures
Conv3	$\frac{(3x3,c)}{p=0} \Rightarrow \frac{(3x3,c)}{p=0} \Rightarrow \frac{(3x3,c)}{p=1} \Rightarrow \frac{(1x1,1)}{p=1} \Rightarrow FC225 \Rightarrow FC92 \Rightarrow O2$
Conv4	$\frac{(3x3,c)}{p=0} \Rightarrow \frac{(3x3,c)}{p=0} \Rightarrow \frac{(3x3,c)}{p=1} \Rightarrow \frac{(3x3,c)}{p=1} \Rightarrow \frac{(1x1,1)}{p=1} \Rightarrow FC225 \Rightarrow FC92 \Rightarrow O2$
Conv5	$\frac{(3x3,c)}{p=0} \Rightarrow \frac{(3x3,c)}{p=0} \Rightarrow \frac{(3x3,c)}{p=1} \Rightarrow \frac{(3x3,c)}{p=1} \Rightarrow \frac{(3x3,c)}{p=1} \Rightarrow \frac{(1x1,1)}{p=1} \Rightarrow FC225 \Rightarrow FC92 \Rightarrow O2$

Table 1 shows these CNN architectures. The number after the *Conv* name represents the depth of the convolutional layers, and it is not included kernel: (1x1, 1). FC stands for fully connected, and the numbers following FC represent the number of neurons. After the kernel (1x1, 1) is processed, followed by flattening, then it is connected to the fully connected layer. O2 represents the output of the two neurons. Finally, we can get the predicted probability of each sudden-death position from the DCNN model.

The big data is essential for any deep learning models. From our previous study, we establish about 12 million positions of board state in the knowledge base, and this study uses a part of it as training set. Table 2 shows the sudden-death positions of Connect6 from move order 2 to 6. The data in the table has been removed from the positions that are stored repeatedly due to the move combination. The Connect6 board is symmetric under horizontal, vertical and diagonal rotation; therefore, all the sudden-death positions in the knowledge base are rotational symmetry.

As can be seen from the table, the proportion of sudden-death positions is quite high. The proportion of M5 is even as high as 26%. In the process of searching, if the sudden-death positions cannot be judged, it is quite unfavorable for the contest of Connect6 game. In view of this, the predictor of sudden-death positions is necessary in the development of the Connect6 game.

TABLE 2. PART OF THE SUDDEN-DEATH POSITIONS OF CONNECT6.

Move Order	Board State	Sudden-Death	Percentage
2	285	46	16.14%
3	80,017	14,846	18.55%
4	780,916	187,278	23.98%
5	793,572	206,984	26.08%
6	779,788	141,649	18.17%

In order to test the ability of the CNN network to identify the sudden-death at the opening of the Connect6 game. This study chooses move order 2 to 5, and there are 409,154 training sets in total. In order to increase the training speed, we selected 100,000 sudden-death positions as the training set. After training, we selected 100 difficult sudden-death positions to test the accuracy in identifying sudden-death positions from different deep CNN architectures.

3.2 Experimental result

Table 3 shows the result of testing deep CNN model. The test data accuracy outside parentheses is from the data of training set, and The numbers in parentheses are additional retained test data, not from the training set. It can be seen from the data that there is little difference in the accuracy rate of the two numbers.

It can be seen from the experimental results that after 4 epochs, an acceptable accuracy has been shown. It is about 80% accuracy. Therefore, it can be seen that using the deep CNN model to predict sudden-death positions is a feasible attempt. Furthermore, the experimental results show that stacking multiple convolutional layers has a greater impact on the prediction of sudden-death positions than the number of filters.

TABLE 3 EXPERIMENTAL RESULT.

epoch / CNN Model	1	2	3	4
Conv3(c=8)	0.47(0.53)	0.52(0.53)	0.50(0.52)	0.67(0.75)
Conv4(c=8)	0.47(0.62)	0.55(0.60)	0.66(0.77)	0.72(0.79)
Conv5(c=8)	0.47(0.64)	0.54(0.77)	0.54(0.76)	0.62(0.84)
Conv3(c=16)	0.18(0.28)	0.18(0.33)	0.52(0.78)	0.86(0.78)
Conv4(c=16)	0.18(0.23)	0.76(0.83)	0.73(0.83)	0.74(0.76)
Conv5(c=16)	0.45(0.53)	0.69(0.67)	0.78(0.79)	0.85(0.82)

4. Conclusion

This study constructs the deep CNN models to predict sudden-death positions of Connect6 game and explores the differences of related deep CNN models from experiments. The construction of this model is of great significance to the design of the Connect6 game, because if the prediction accuracy of the model is high, the number of candidate moves during the search can be greatly reduced, which will help improve the search efficiency. We summarize the contributions and the conclusions of the study as follows.

- An experimental study of different CNN-based architectures of deep CNN models for Connect6.
- In Connect6 game, this study proposes using deep convolutional neural network to predict the sudden-death positions, and it improve the search performance of Kavalan, which is an AI program we design to play Connect6 game.
- The experimental results show that the depth of stacking multiple convolutional layers has a greater impact on the prediction of sudden-death positions than the number of filters.

With the rise of artificial intelligence (AI), people need the ability to think and judge logically. Puzzle games can be used as a tool for training logic, which has a positive effect on the sustainable development of society. Connect6 is a very interesting puzzle game; therefore, the deep CNN model developed in this study is of great significance in the research of Connect6 game.

5. future work

This study builds the sudden-death positions model of Connect6, and uses a large amount of data in the knowledge base for deep learning, in order to find the best model to predict the sudden-death positions of Connect6. There are still some unfinished parts of this study; therefore, the future works of this study are as follows:

1. This study only trains the deep CNN model for the positions of move order 2 to 5. In the future, it can be expanded to all sudden-death positions in the knowledge base, so that the trained model will be more in line with the purpose of this study. Furthermore, predictions for sudden-death positions will also be more accurate.
2. The kernel size selected in this study is only 3x3. In the future, experiments can be carried out on kernels of different sizes to understand the accuracy of different sizes of kernels and the difference in learning efficiency. This has important implications for building deep CNN model to predict sudden-death positions.

References

[1] ConvNetSharp, https://github.com/cbovar/ConvNetSharp.

[2] Gao, Chao, Ryan Hayward, and Martin Müller. "Move prediction using deep convolutional neural networks in Hex." IEEE Transactions on Games 10.4 (2017): 336-343.

[3] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." Proceedings of the fourteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2011.

[4] Jarrett, Kevin, et al. "What is the best multi-stage architecture for object recognition?." 2009 IEEE 12th international conference on computer vision. IEEE, 2009.

[5] Khan, A., Sohail, A., Zahoora, U. et al. "A survey of the recent architectures of deep convolutional neural networks," Artif Intell Rev 53, 5455–5516 (2020).

[6] Liskowski, Paweł, Wojciech Jaśkowski, and Krzysztof Krawiec. "Learning to play othello with deep neural networks." IEEE Transactions on Games 10.4 (2018): 354-364.

[7] Silver, D., Huang, A., Maddison, C. et al, “Mastering the game of Go with deep neural networks and tree search,” Nature 529, 484–489 (March, 2016).

[8] Silver, D., Schrittwieser, J., Simonyan, K. et al., “Mastering the game of Go without human knowledge,” Nature 550, 354–359 (October, 2017).

[9] ″Softmax function.” Wikipedia, Wikimedia Foundation, 5 October 2022, https://en.wikipedia.org/wiki/Softmax_function.

[10] Tan, Hong Hui, and King Hann Lim. "Vanishing gradient mitigation with deep learning neural network optimization." 2019 7th international conference on smart computing & communications (ICSCC). IEEE, 2019.

[11] Wu, I-Chen, Dei-Yen Huang, and Hsiu-Chen Chang. "Connect6." ICGA Journal 28.4 (2005): 235-242.

[12] Wu, I-Chen, and Ping-Hung Lin. "Relevance-zone-oriented proof search for connect6." IEEE Transactions on computational intelligence and AI in games 2.3 (2010): 191-207.

[13] Yang, Jung-Kuei and Yen, Shi-Jim, "Conservative Relevance Zone Search in Connect6," 2011 International Conference on Technologies and Applications of Artificial Intelligence, 2011, pp. 273-279, doi: 10.1109/TAAI.2011.65.

[14] Yen, Shi-Jim, and Yang, Jung-Kuei. "Two-stage Monte Carlo tree search for Connect6." IEEE Transactions on Computational Intelligence and AI in Games 3.2 (2011): 100-118.

[15] Yen, Shi-Jim, and Yang, Jung-Kuei, Kao Kuo-Yuan and Yang Tai-Ning, "Bitboard knowledge base system and elegant search architectures for Connect6," Knowledge-based systems 34 (2012): 43-54.