A Hybrid Gomoku Deep Learning Artificial Intelligence

Peizhi Yan
Lakehead University
955 Oliver Rd
Thunder Bay, Ontario, Canada P7B 5E1
+1 705-943-0919
pyan@lakeheadu.ca

Yi Feng
Algoma University
1520 Queen Street East
Sault Ste. Marie, Ontario, Canada P6A 2G4
+1 705-949-2301, ext. 4315
yi.feng@algomau.ca

ABSTRACT

Gomoku is an ancient board game. The traditional approach to solving the Gomoku is to apply tree search on a Gomoku game tree. Although the rules of Gomoku are straightforward, the game tree complexity is enormous. Unlike other board games such as chess and Shogun, the Gomoku board state is more intuitive. This feature is similar to another famous board game, the game of Go. The success of AlphaGo [5, 6] inspired us to apply a supervised learning method and deep neural network in solving the Gomoku game. We designed a deep convolutional neural network model to help the machine learn from the training data. In our experiment, we got 69% accuracy on the training data and 38% accuracy on the testing data. Finally, we combined the trained deep neural network model with a hard-coded convolution-based Gomoku evaluation function to form a hybrid Gomoku artificial intelligence (AI) which further improved performance.

CCS Concepts

• Computing methodologies → Supervised learning
• Computing methodologies → Planning under uncertainty

Keywords

Gomoku; convolution; convolutional neural network; deep learning; supervised learning; artificial intelligence; evaluation function

1. INTRODUCTION

1.1 The Gomoku Game

The Gomoku game was derived from the game of Go. It uses a game board with 15x15 grid intersections. Like the game of Go, two players use white and black stones respectively to play the game. At each turn, the current player can put only one stone on one of the unoccupied spaces. The first player to form an unbroken chain of five stones in a row (see Figure 1) is the winner. In other words, if any of those patterns in Figure 1 appear on the game board, the player who formed those patterns wins. The unbroken chain of five stones could be diagonal, vertical, or horizontal. Therefore, Gomoku is also called five in a row.

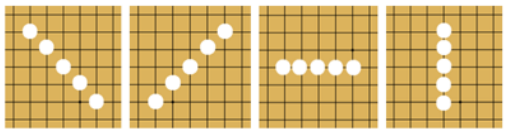
Figure 1. Example of four possible winning patterns.

Table 1. Some Board Game Complexities

Game	State Space Complexity	Game Tree Complexity	Reference
Gomoku	10^{105}	10^{70}	[1]
Chess	10^{47}	10^{123}	[2]
Go	10^{170}	10^{360}	[3]

The state space complexity of Gomoku is about 10^{105}, whereas the game tree complexity of Gomoku is about 10^{70} (see Table 1). Most Gomoku AI agents use a game tree search algorithm. A good Gomoku AI agent must have a well-designed game state evaluation function and a machine with enough computing power to perform a more in-depth tree search. The challenge in designing a sophisticated evaluation function is that it requires the participation of Gomoku game experts to assess the possible patterns.

1.2 Deep Neural Network

A deep neural network (DNN) is an artificial neural network with more than one layer between the input layer and the output layer. Convolution layers are normally applied right after the input layer. This kind of model is called a deep convolutional neural network (deep CNN) [9]. There are many trainable filters in each convolution layer that can capture some significant visual patterns in the input. The deep CNN can make high-level abstractions from the input [14]. These high-level abstractions may be perplexing to human beings, but are helpful to the CNN to recognize the connection between the visual patterns in the input. Therefore, in image classification, a deep CNN has many advantages [4].

1.3 Value Gradient

The value network used in AlphaGo [5, 6] generates a single value as the evaluation of the game for any input Go game state. The value network of AlphaGo is used in a Monte Carlo Tree Search [8]. In this paper, we did not use any game tree search algorithm to evaluate each possible move in the game tree with a given depth. Instead, we use a single two-dimensional matrix as a value gradient to guide our Gomoku AI to make decisions. Each value at a position in the value gradient represents the value that would determine whether to make a move at that position.

The rest of this paper is structured as follows. In Section 2 we give the algorithm for the Gomoku game state evaluation function. In Section 3 we describe the deep learning method we used to improve the performance of the evaluation function mentioned in Section 2. We call the new Gomoku AI a hybrid Gomoku AI. Section 4 presents the experimental results of this hybrid Gomoku AI. The final section is our conclusions.

2. CONVOLUTION-BASED GOMOKU EVALUATION ALGORITHM

Instead of using many if-else statements, we used two-dimensional convolution [15] to detect some basic patterns in the input Gomoku game state. One of the prerequisites of this algorithm is the pre-defined filters. Each filter contains a visual pattern. In our experiment, we use only 16 filters to detect 16 different patterns (see Figure 2). We use a 15×15 two-dimensional matrix of integers to encode the Gomoku game state (we assume the human player uses white stones), for each position: 0 represents an unoccupied space; 1 represents the white stone; -1 represents the black stone. We also need an integer matrix of the same size as the encoded Gomoku game state as the value gradient.

2.1 Convolution Operation

We apply full two-dimensional convolution operations to the game state matrix (with zero padding) with each filter of size $a \times a$. The size of the output matrix is: $b \times b$, where $\mathrm{b} = 15 + (\mathrm{a} - 1)$. When the filter detects the same visual pattern in the game state, it will generate a value γ which is the number of non-zero values in this filter. By counting the number of values that are equal to γ in the output feature map, we can determine the number of appearances of that visual pattern in the game state (see Figure 3).

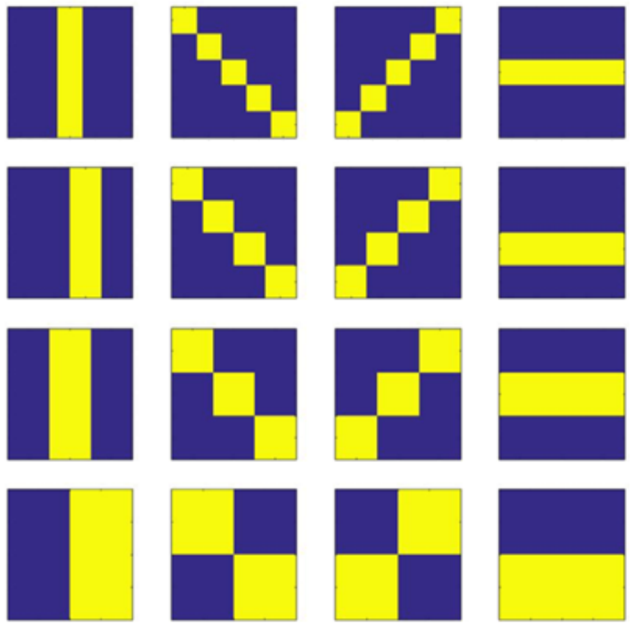

Figure 2. The set of filters in our algorithm. The first row of filters are 5×5 matrices, the second row of filters are 4×4 matrices, the third row of matrices are 3×3 matrices, the last row of filters are 2×2 matrices. In this figure, blue represents the value zero and yellow represents the value one.

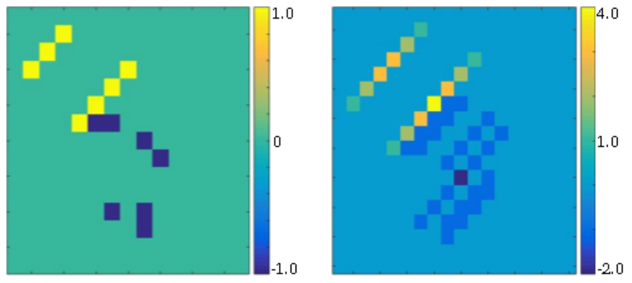

Figure 3. The left sub-figure is the example of an encoded Gomoku game state. The right sub-graph is the output feature map after the convolution operation with the third filter in the second row of Figure 2.

2.2 The Evaluation Algorithm

We use integers x and y to represent row index and column index. The encoded game state matrix is B, the value gradient matrix is V. We also need a set of filters $\psi = \{F_1, F_2, F_3, \ldots, F_n\}$, a set of weights $\omega = \{w_1, w_2, w_3, \ldots, w_n\}$. Each weight in ω is associated with a corresponding filter in ψ. A higher weight represents that the pattern indicated in the corresponding filter has higher value on the game board. The steps of this algorithm are described as follows:

Step 1: Initialize V with zeros. Create a matrix that has the same size as B, called B', each value in B' should be the negative of the corresponding value in B ($B'_{x,y} = -B_{x,y}$).

Step 2: Loop through all the positions in B (in total 15×15=225 positions). For each occupied position ($B_{x,y} \neq 0$), set the value of $V_{x,y}$ to 0. For each unoccupied position ($B_{x,y} = 0$), do step 3. When the iteration is complete, stop. V is the final value gradient matrix.

Step 3: Set the value of both $B_{x,y}$ and $B'_{x,y}$ to 1. For each filter $F_i \in \psi$, do the two-dimensional convolution operation with B, make the output feature map M. Make the number of 1s in F_i as γ (for example, if there are three 1s in F_i, then γ=3). Count the number of the values in M that are equal to γ, then multiply this number by w_i, make the result v_i. Do the same thing to B', make the result v'_i. Change both $B_{x,y}$ and $B'_{x,y}$ to 0. The final value of position (x, y) is as shown below (α is the weighting coefficient between 0 and 1, we use 0.5 in this paper).

$$V_{x,y} = \alpha \times \sum_i \mathrm{v}_i + (1 - \alpha) \times \sum_i \mathrm{v'}_i$$

3. HYBRID GOMOKU AI

In our experiment, we use a specially designed DNN as our machine learning model. The original dataset is the RenjuNet[1] dataset. We preprocessed the original dataset to get our training and testing datasets. There are 414674 labeled Gomoku states in the training set and 104255 labeled Gomoku states in the testing set. We combined the output of our neural network model with the output of the Gomoku AI discussed in Section 2, to get a hybrid value gradient which can lead the Gomoku AI agent make move predictions for any given Gomoku state. In this section, we describe the dataset preprocessing method, the DNN model, and the implementation of our hybrid Gomoku AI.

[1] http://www.renju.net/downloads/downloads.php

3.1 Data Preprocessing

The encoding of the Gomoku game state was discussed in Section 2. The original RenjuNet dataset we used had 53183 Gomoku game records. Each game record is a sequence of moves. A move is represented by the position on a Gomoku game board. We use the computer to simulate those games to get the encoded Gomoku game state. It is worth noting that we did not keep the order of black and white. Instead, we inverted the stones' color (1 will be -1, -1 will be 1) on the game state after each move. Therefore, each next move is equivalent to putting a white stone onto the game board. In each preprocessed game state, the white stones are "ours," and the black stones are the "enemy's." By doing this, we reduced the learning complexity. In contrast, AlphaGo's dataset kept the order of black and white, and this increased the difficulty of learning. Furthermore, we only kept the first 15 states in each game record. After preprocessing, we had 518929 labeled Gomoku states. The ratio of training data to testing data is about 4:1.

3.2 DEEP NEURAL NETWORK ARCHITECTURE

As shown in Figure 4, the DNN architecture in this paper contains two sub-DNNs. One of the sub-DNNs is a CNN, and the other is a deep feedforward network. Both sub-DNNs work in parallel, which means that the input will go to both sub-DNNs at the same time, and the output is the matrix addition of the results of both sub-DNNs.

In the CNN section, the first convolution layer (conv layer 1) has 64 3x3 filters of stride 1. The input to this layer is the15x15 encoded Gomoku state. The activation function is the rectified linear unit (ReLU) [7]. The output has 64 13x13 feature maps. The second convolution layer (conv layer 2) has 32 3x3 filters of stride 1. The activation function is ReLU. The output has 32 11x11 feature maps. This layer is fully connected to a fully connected layer (fc layer 1) which has 1024 neurons. It fully connects another fully connected layer (fc layer 2) which has 512 neurons. The second fully connected layer (fc layer 2) is fully connected with the third fully connected layer (fc layer 3) which has 225 neurons. We use this CNN to capture the stone patterns on the input Gomoku state.

In the other sub-DNN, there are only two fully connected layers (fc layer *a* and fc layer *b*). Fc layer *a* has 512 neurons; fc layer *b* has 225 neurons. All the fully connected layers in this paper use the ReLU activation function. We apply softmax [16] to the output first. Then the output will be reshaped to a 15x15 matrix. The idea of using this sub-DNN came from the residual networks [10, 11]. The softmax function is defined as:

$$\sigma(z)_j = e^{z_j} / \sum_{i=1}^{225} e^{z_i} \text{ , for j=1, 2, ..., 225}$$

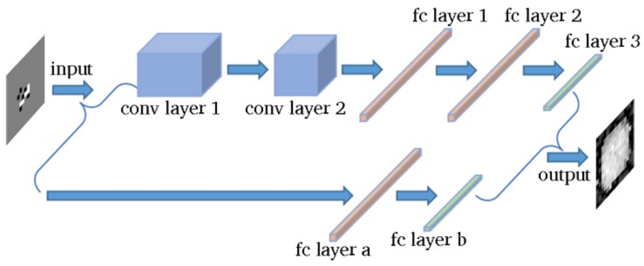

Figure 4. Architecture of the DNN used in our paper. The upper sub-DNN is a CNN, the lower sub-DNN is a deep fully connected network.

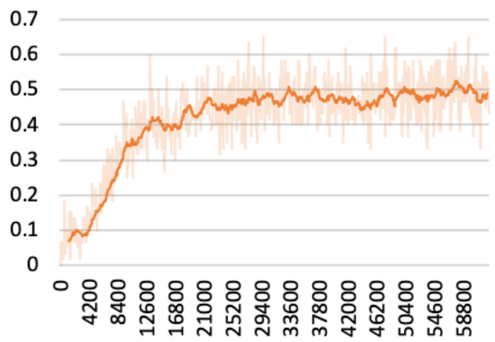

Figure 5. The batch accuracy during 10 training epochs. The horizonal axis represents the training steps (each training step consists of a batch of training data); the vertical axis represents the batch accuracy (range from 0.0 to 1.0). The darker orange line is the average batch accuracy value, the lighter orange line is the actual batch accuracy value.

3.3 Training the Model

The loss function we used in this paper is the cross-entropy loss [12]. We use the Adam optimizer [13] to optimize our model. We set the learning rate to a constant: 0.001. The batch size of our training data is 60. We trained our model for ten epochs (see Figure 5), each epoch consists of a full training cycle on the training set). The prediction accuracy on the training dataset is 69%; the prediction accuracy on the testing dataset is 38%.

3.4 Hybrid Gomoku AI

Since the output of our DNN model is a matrix of predicted probabilities of next moves, we use it as a value gradient. The sum of all the values in the value gradient made by our DNN model is 1, we want to scale those values with respect to the value gradient generated by the simple convolution-based Gomoku AI. Produce the value gradient generated by our model V'; produce the value gradient generated by the convolution-based Gomoku AI V. We first sum the values in V, calling it *s*:

$$s = \sum_x \sum_y V_{x,y}$$

Then we multiply each value in V' by s (if s > 0):

$$V'_{x,y} = s \times V'_{x,y} \text{ , if s > 0}$$

Finally, we add the two value gradient matrices together, to get the hybrid value gradient ($V''_{x,y}$):

$$V''_{x,y} = V_{x,y} + V'_{x,y}$$

In this paper, we only use the hybrid Gomoku AI in the first 15 moves of each game, that is because we only use the training data with the first 15 states in each game record to train the DNN model.

4. EXPERIMENTAL RESULTS

We did not remove duplicate samples (the duplicate samples have same Gomoku state but could have different labels) from the training dataset in this paper. The primary reason is that the original RenjuNet Gomoku game records were not guaranteed to come from Gomoku experts' competition. Also, there could be more than one suitable next moves for a given Gomoku state. If we remove the duplicate samples, we may lose many appropriate alternative moves. We experimented with training the same DNN for the same amount of training steps, but use the training dataset without duplicate samples. The result proved our point of view (Table 2).

Table 2. Training and Testing Accuracy

Dataset	Number of Samples	Training Accuracy	Testing Accuracy
With duplicate	414674	69%	38%
Without duplicate	129164	49%	32%

Because the distribution of labels in the training dataset is highly imbalanced (see Figure 6) [17], our model easily captured this feature and went to a local optimum. As training progressed, this effect diminished but still remained. (see Figure 7). We visualized the output value gradients of both the simple hard-coded Gomoku AI and the hybrid Gomoku AI (see Figure 8).

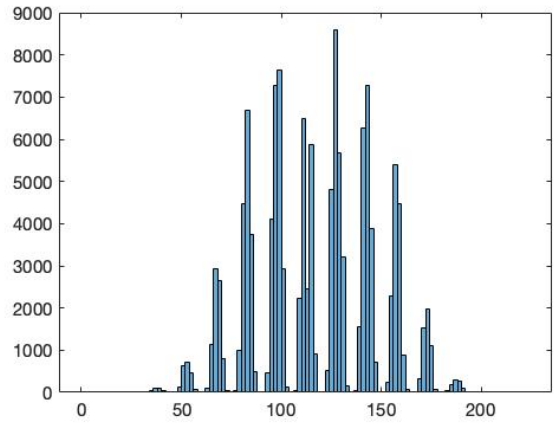

Figure 6. The plot of the distribution of labels in the training dataset. The horizontal axis is label; the vertical axis is the number of occurrence of the labels in our training dataset.

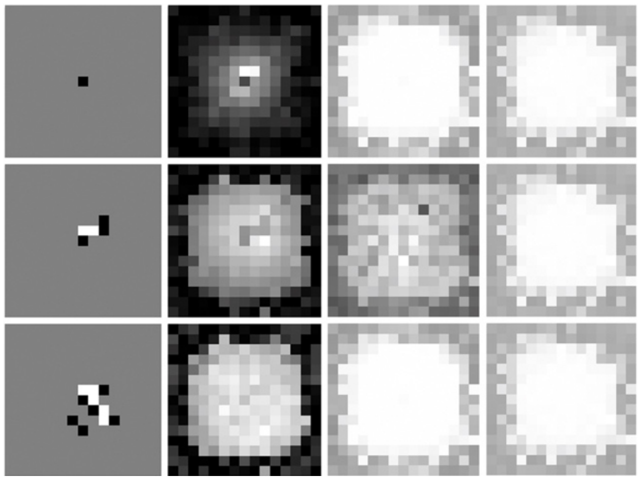

Figure 7. The first (leftmost) column of images are some example Gomoku states in our training dataset. Other images are the visualization of the output. Each row of outputs represent the same input image but in different training stages. The second column of outputs were generated by our trained model; the third column of outputs were generated by the model after 6900 training steps; the last (rightmost) column of outputs were generated by the model after 500 steps of training.

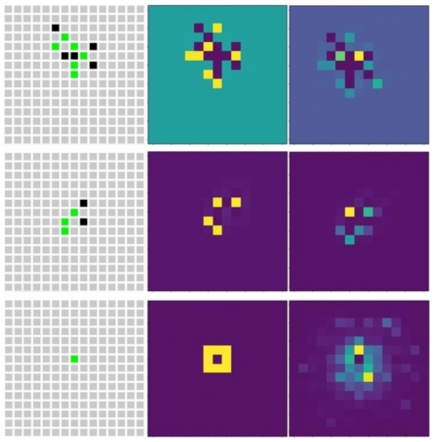

Figure 8. Visualized value gradients on some example input Gomoku states. Images in the first column are input Gomoku states (green stones are AI's, black stones are human player's). Images in the second column are the visualization of value gradients generated by the simple hard-coded Gomoku AI. Images in the third column are the visualization of value gradients generated by the hybrid Gomoku AI. A darker color (purple) represents a lower value; a lighter color (yellow) represents a higher value

We also did another interesting experiment. We used only six predefined 3x3 filters (see Figure 9) in the first convolution layer, and use 128 3x3 filters in the second convolution layer. The first convolution layer is untrainable. The pre-defined filters can capture the status of connected stones on a Gomoku state. The second convolution layer should learn some abstract connections from the output of the first convolution layer. With the help of the predefined filters, we speed up the training process. We got the same training and testing accuracy within seven epochs. In Figure 10, we compared the batch accuracy during the learning process of the DNN model with predefined filters (red lines) and the original DNN model (blue lines) we used in Section 3.

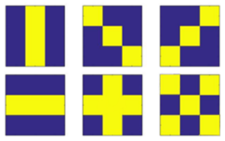

Figure 9. Six predefined filters (blue represents 0; yellow represents 0.1).

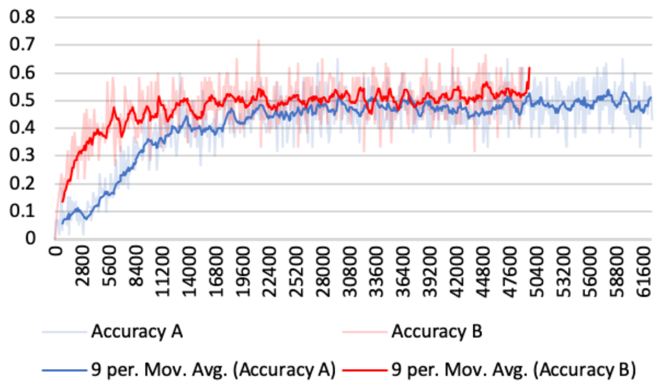

Figure 10. Vertical axis is the accuracy; horizontal axis is the training step. Moving average A and B use 9000 steps (100 steps between 2 nearest data points) as a period.

5. CONCLUSION

Although the problem in this paper seems like an image classification problem, it is different from the other image classification problems:

1) In an image classification problem, the impact of a single pixel is not as significant as the impact of a single pixel on our Gomoku state. The reason is that each pixel on a Gomoku state represents a stone, a single stone could have a significant impact on the value of the Gomoku state. That is why the filters in this DNN are smaller than the filters used in many other CNN image classifiers.
2) The labels in our training dataset are highly unbalanced (see Figure 6). We cannot just remove some of the data to balance the training dataset since this will waste many good training examples.
3) Duplicate images in an image classification problem should be removed. However, we kept the duplicate training data. Because for a given Gomoku state, there could be more than one good next move.

In the CNN of our DNN, the input to the first convolution layer is the Gomoku state, and more filters could learn more patterns from the input Gomoku state. The output of the first convolution layer will be the abstraction of the Gomoku state, the filters in the second convolution layer should learn the connection between the patterns. The purpose of the sub-DNN which is a deep feed-forward neural network is to reduce the impact of the distortion of Gomoku state in the CNN. Besides, since we should not make a move on the occupied positions, the deep feed-forward neural network should learn this feature and constrain the output of the CNN.

By predefining the filters in the first convolution layer using our knowledge of the Gomoku game, we successfully led the CNN to learn the training data in a predefined way, which is faster than training a CNN with randomly initialized filters.

6. ACKNOWLEDGMENTS

The first author would like to thank Dr. Yi Feng and Dr. George Townsend for their contributions to this paper.

7. REFERENCES

[1] Louis Victor Allis. 1994. *Searching for solutions in games and artificial intelligence*. Ponsen & Looijen, Wageningen.

[2] Claude E. Shannon. 1988. Programming a Computer for Playing Chess. *Computer Chess Compendium*: 2–13. http://doi.org/10.1007/978-1-4757-1968-0_1.

[3] John Tromp and Gunnar Farnebäck. 2007. Combinatorics of Go. *Computers and Games Lecture Notes in Computer Science*: 84–99. http://doi.org/10.1007/978-3-540-75538-8_8.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Communications of the ACM* 60, 6: 84–90. http://doi.org/10.1145/3065386.

[5] David Silver, Aja Huang, Chris J. Maddison, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587: 484–489. http://doi.org/10.1038/nature16961.

[6] David Silver, Julian Schrittwieser, Karen Simonyan, et al. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676: 354–359. http://doi.org/10.1038/nature24270.

[7] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning - ICML 10*: 807–814.

[8] Chaslot, Guillaume, Bakkes, et al. 2018. Monte-Carlo Tree Search: A New Framework for Game AI. *AIIDE*.

[9] Yann Lecun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. 1999. Object Recognition with Gradient-Based Learning. *Shape, Contour and Grouping in Computer Vision Lecture Notes in Computer Science*: 319–345. http://doi.org/10.1007/3-540-46805-6_19.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. http://doi.org/10.1109/cvpr.2016.90.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. Computer Vision – ECCV 2016 *Lecture Notes in Computer Science*: 630–645. http://doi.org/10.1007/978-3-319-46493-0_38.

[12] Shie Mannor, Dori Peleg, and Reuven Rubinstein. 2005. The cross entropy method for classification. *Proceedings of the 22nd international conference on machine learning - ICML 05*: 561–568. http://doi.org/10.1145/1102351.1102422.

[13] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv*. Retrieved from https://arxiv.org/abs/1412.6980.

[14] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553: 436–444. http://doi.org/10.1038/nature14539.

[15] Steven W Smith. 1997. *The scientist and engineer's guide to digital signal processing*. California Technical Pub., San Diego.

[16] Nasser M Nasrabadi. 2007. Pattern Recognition and Machine Learning. *Journal of Electronic Imaging* 16, 4: 049901. http://doi.org/10.1117/1.2819119.

[17] Haibo He and E.a. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* 21, 9: 1263–1284. http://doi.org/10.1109/tkde.2008.239.

Authors’ background

Your Name	Title*	Research Field	Personal website
Peizhi Yan	master student	Machine learning and computer vision	https://peizhiyan.github.io/
Yi Feng	associate professor	Formal hardware verification, computer system design	

*This form helps us to understand your paper better, the form itself will not be published.

*Title can be chosen from: master student, Phd candidate, assistant professor, lecture, senior lecture, associate professor, full professor