The discussion of the effect of residual connections in the Convolutional Neural Network for the self-play of Gomoku

Rui Wang (rw22784)

Dec 9, 2019

Link for codes

https://drive.google.com/open?id=1UVrjAit9lHjzjAyEI0C_3jWnMZDrfo1s

1) run trainRes.ipynb to train the models. The trained results has been saved in the ./saved_models
2) run draw_results.ipynb to draw Figure 5 and 6

Link for video

https://www.youtube.com/watch?v=tfxyfgNWeXo

1 Introduction

The Convolutional Neural Networks (CNNs) have been generally applied in the pattern recognition tools in the image classification [1-2], object detection/localization [3-4] and image segmentation [5] tasks. Compared with the fully connected deep neural networks, the CNNs have much less parameters to train due to the properties of weight sharing and local connection. The AlexNet [6] applied the ReLU activation function to suppress the gradient diminishing and obtain faster training efficiency. It also applies the dropout and data augmentation to suppress the overfitting. The VGGnet [7] later applies the small convolutionary kernel (3×3) and max pooling kernel (2×2) to increase the nonlinearity and decrease the number of parameters. The appropriate weight initialization and batch normalization are also proposed. The GoogLeNet [8] applies the multiple branches with different convolutionary kernel sizes and realizes deeper CNNs. The ResNet [9] applies the shortcut connection which is in parallel with the serial of conv/batch norm/ReLU layers, suppresses the gradient diminishing and obtains better performances than the CNNs without such residual connections in the image classification tasks. However, the drawback is that the ResNet is more difficult to train than the conventional CNNs.

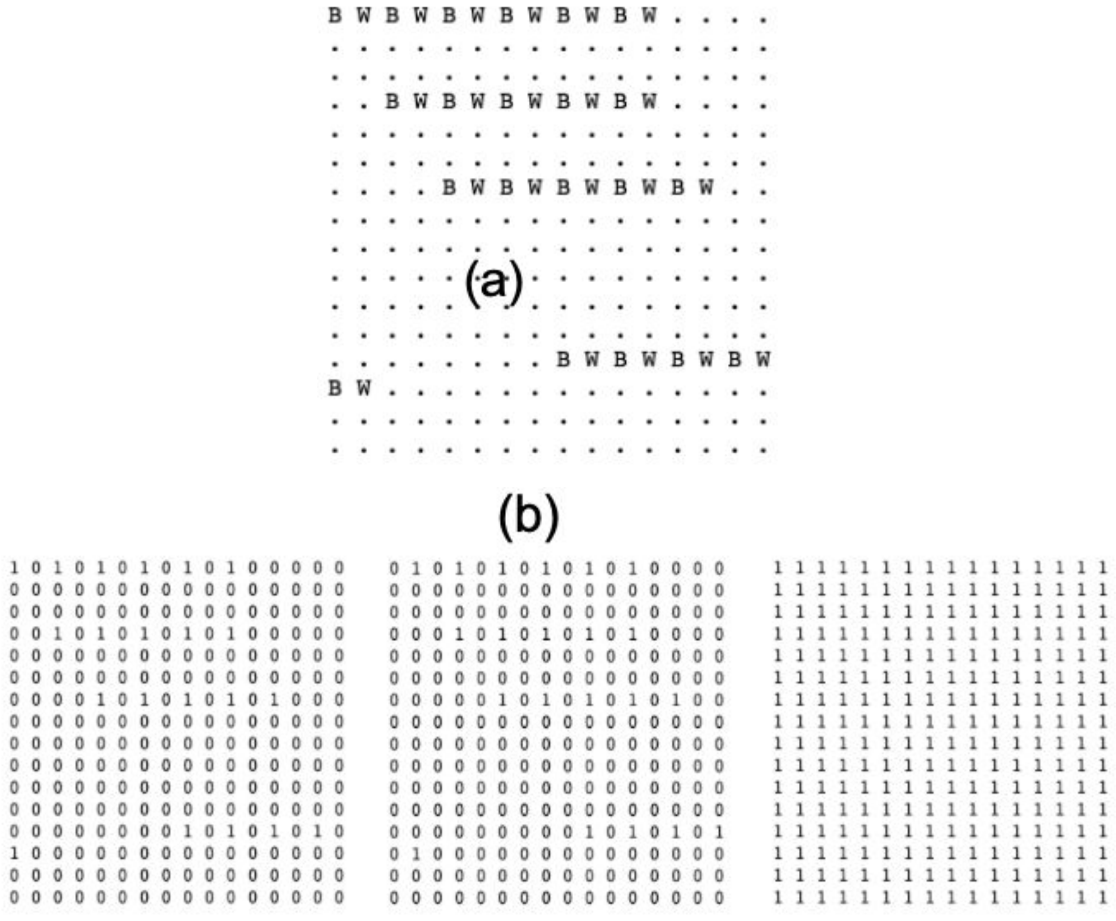
The CNNs are also applied in the self-play of Gomoku by AlphaGo [10-11]. In the training of the policy and value networks of the self-play model of Gomoku, the AlphaGo did not apply the fully connected neural network which should take the input size as 256, i.e., 16×16. Instead, the model regards the chess as virtual 16×16 images with discreet pixel values. The model treats the policy and value calculations as the pattern recognition problems of the virtual image. The AlphaGo applies the residual connections in such CNNs.

Although the role of the residual connections in the computer vision tasks have been generally discussed, the effect of the residual connections on the performances of the Gomoku self-play models as a reinforcement learning tasks has seldom been discussed. In this report, I trained a self-play model of Gomoku with the CNNs with and without residual connects and compare the performances in two ways: (1) compare the loss of the policy and value networks of the two models, (2) let the two trained models play with each other and collect the statistics of the winner. The experiment shows the obvious positive effect of the residual connects in such a reinforcement learning task. Note that the comparison is fair because the two types of CNNs are with the same number of parameters to train.

2 Methodology

2.1 chess representations as three-channel images

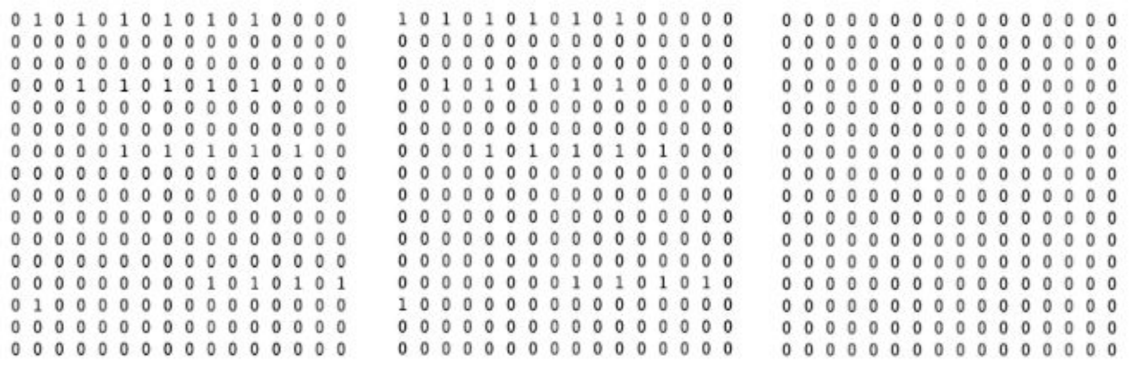
Each chess is represented as a three-channel image. The pixel values are binary: 0 and 1. When the palyer is Black (White), the third channel is filled with 1 (0). For a certain player, the first channel is the distribution of his stones represented as 1 (with 0 as the background), the second channel is the distribution of his energy’s stones also represented as 1. Figure 1 illusttrates the process.



```
B W B W B W B W B W B W . . . .
. . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
. . B W B W B W B W B W . . . .
. . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
. . . . B W B W B W B W B W . .
. . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
. . . . . . . . B W B W B W B W
B W . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
```

```
1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0    0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0    0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0    0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0    0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

(c)



```
0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0    1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0    0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0    0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1    0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0    1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 1. The representation of the chess with three-channel images. (a) is the original chess. (b) and (c) are the three channels of the virtual images when the player is Black and White, respectively.

2.2 policy and value network structures

Figure 2 and 3 show the policy and value network structures with and without residual connections, respectively. The two cases have the same number of parameters to train. There are two residual or non-residual blocks. The action is defined as putting the stone onto one of the 256 positions. For each training instance, the policy network outputs a 256-dim vector which means the probability of putting the stone onto the corresponding position. The value network outputs the action value which is a scalar. The policy network output is activated with softmax. The value network is activated with tanh.

The loss function of the policy network is defined as

$$Loss_{policy} = (1/m) \sum_{batch=1}^{batch=m} \sum_{k=1}^{256} - \widehat{P_k} In\,(P_k)$$

The Loss function of the value network is defined as

$$Loss_{value} = (1/m) \sum_{batch=1}^{batch=m} \left(\widehat{\widehat{V}} - V\right)^2$$

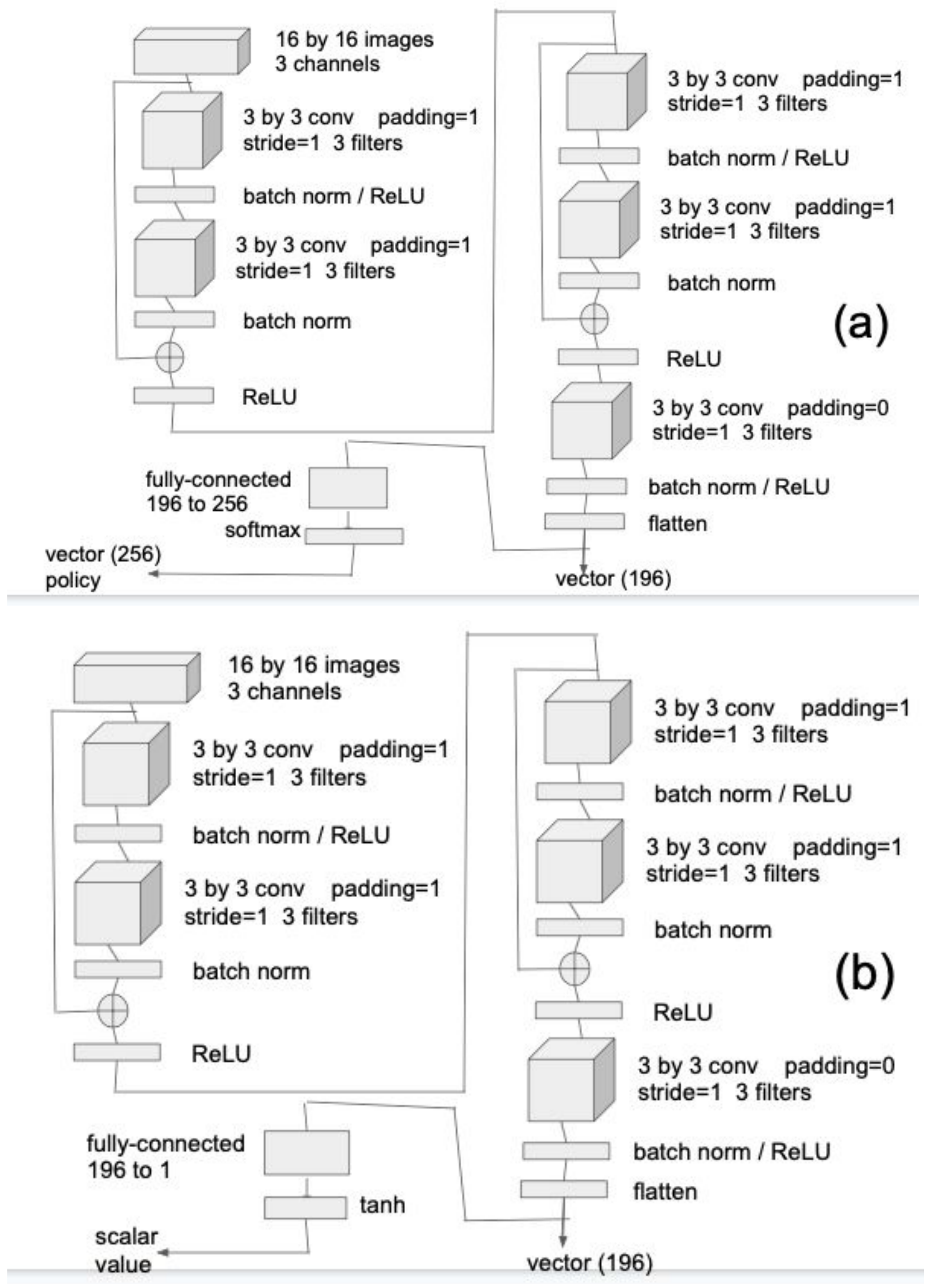


Figure 2. (a) The policy and (b) value networks with residual connections.

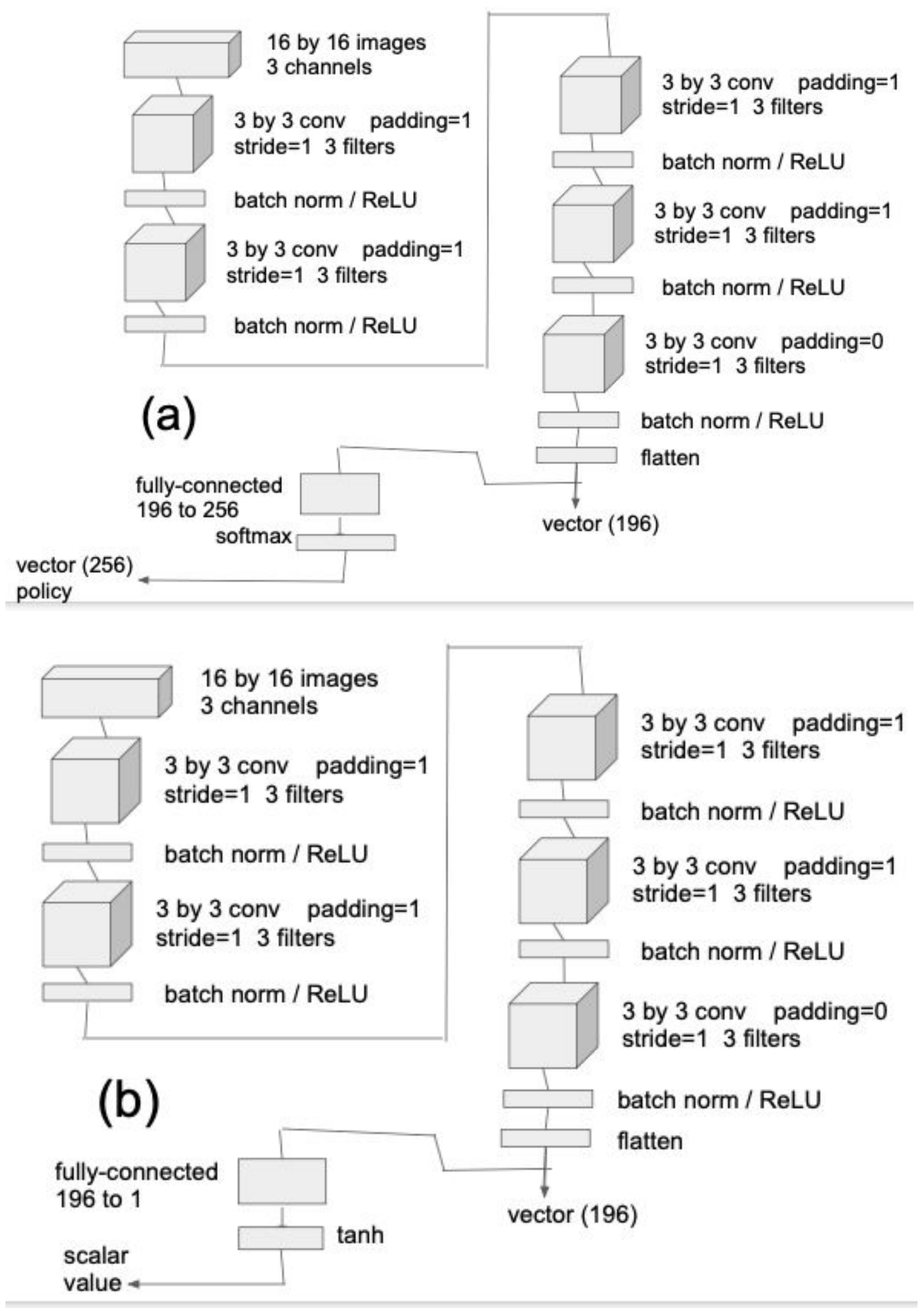
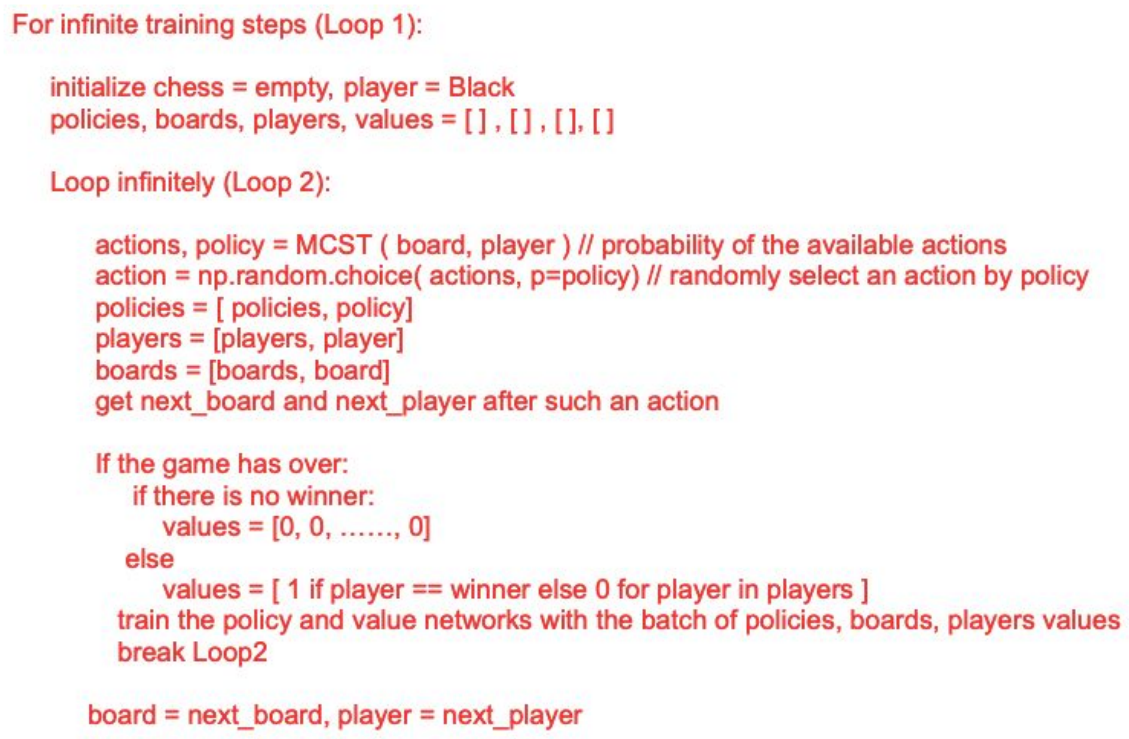


Figure 3. (a) The policy and (b) value networks without residual connections.

2.3 training process

Figure 4 shows the training process based on the reinforcement model. Given a chess pattern, the Monte Carlo Search Tree (MCST) is applied to get the probabilities of the available actions. The MCST model in [11] is applied.

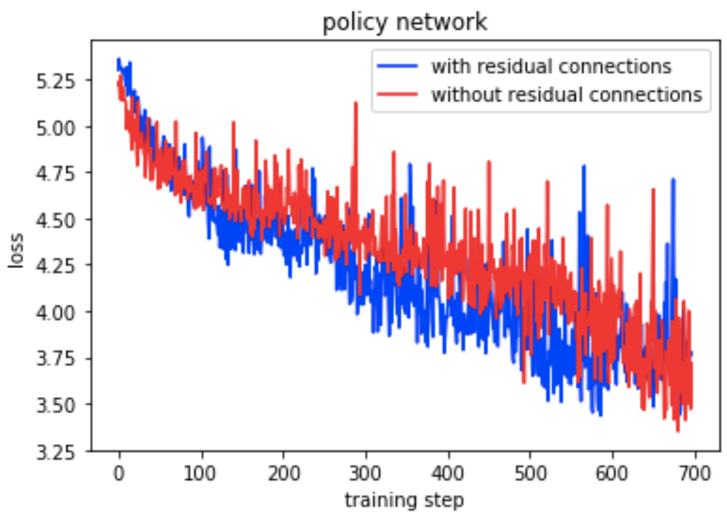


```
For infinite training steps (Loop 1):

    initialize chess = empty, player = Black
    policies, boards, players, values = [ ] , [ ] , [ ], [ ]

    Loop infinitely (Loop 2):

        actions, policy = MCST ( board, player ) // probability of the available actions
        action = np.random.choice( actions, p=policy) // randomly select an action by policy
        policies = [ policies, policy]
        players = [players, player]
        boards = [boards, board]
        get next_board and next_player after such an action

        If the game has over:
            if there is no winner:
                values = [0, 0, ……, 0]
           else
                values = [ 1 if player == winner else 0 for player in players ]
          train the policy and value networks with the batch of policies, boards, players values
          break Loop2

        board = next_board, player = next_player
```

Figure 4 The training process of the reinforcement learning model

3, Results the Discussions

Figure 5 shows the evolution of the training loss of the policy and value networks with the training steps. The training loss of the networks with the residual connections converge faster than those without the residual connections. However, the residual connections lead to more non-stability of the value networks. Aa a comparison, the value networks without the residual connections are hard to converge.



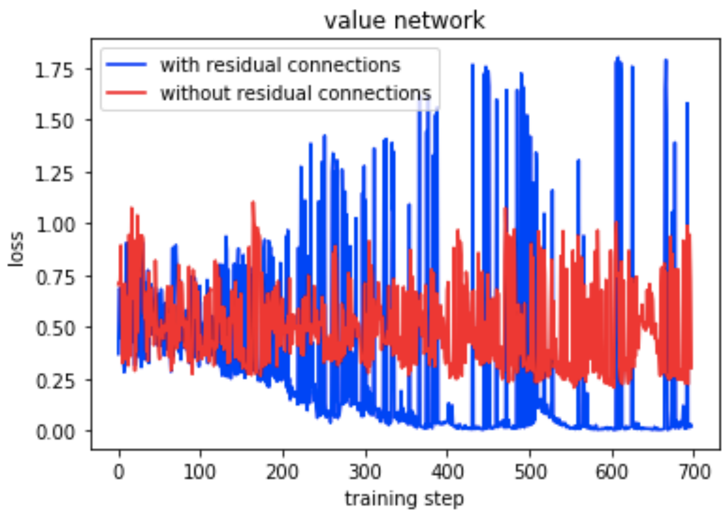


Figure 5. The evolution of the training loss of the policy and value networks with the training steps

In the following the effect of the residual connections are further illustrated via the beating of the two trained models at different training steps. The model with (without) the residual connections uses the black (white) stones. As Figure 6 shows, at different training steps, the trained models with the residual connections win in most of the times.

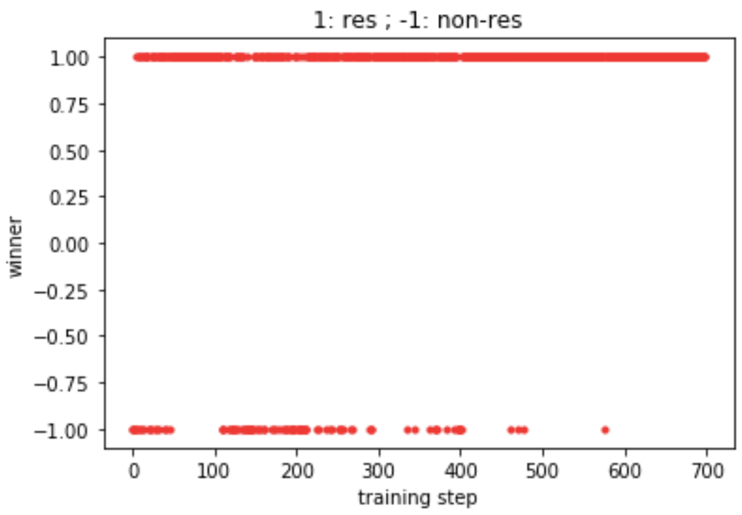


Figure 6. The evolution of winner with the training step in the beating of the trained models with and without residual connections.

4 Conclusion

I for the first time illustrate the effect of the residual connections in the CNNs of the self-play of Gomoku based on the policy gradient model. The residual connections help the training loss of the policy and value networks converge faster. However, at the same time, it on the other hand increases the non-stability of the value network. In the beating games of the trained models with and without residual connections, the former model wins in most of the time, which demonstrates the advantage of the application of residual connections in this task.

References

[1] Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, Jürgen Schmidhuber, “Flexible, High Performance Convolutional Neural Networks for Image Classification, ” AAAI 2011.
[2] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, Zheng Zhang, “The Application of Two-Level Attention Models in Deep Convolutional Neural Network for Fine-Grained Image Classification, ” CVPR 2015.
[3] Sachin Sudhakar Farfade , Mohammad J. Saberian, Li-Jia Li, “Multi-view Face Detection Using Deep Convolutional Neural Networks, ” ICMR 2015.
[4] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, Gang Hua, “A Convolutional Neural Network Cascade for Face Detection, ” CVPR 2015.
[5] Fausto Milletari ; Nassir Navab ; Seyed-Ahmad Ahmadi, “V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation, “ 3DIMPVT 2016.
[6] Krizhevsky A, Sutskever I, Hinton G E, “ImageNet classification with deep convolutional neural networks,” International Conference on Neural Information Processing Systems, 2012.
[7] Karen Simonyan, Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” CVPR 2015.
[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, “Going Deeper with Convolutions, ” CVPR 2014.
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep Residual Learning for Image Recognition, ” CVPR 2015.
[10] Zheng Xie, XingYu Fu, JinYuan Yu, “AlphaGomoku: An AlphaGo-based Gomoku Artificial Intelligence using Curriculum Learning,” arxiv, 2018.
[11] https://github.com/zhixiangli/alphazero-gomoku