Csc 665 final project report

Xiaowei Xu

May 11, 2019

1 Abstract (you can skip reading this part ...)

The Google's AlphaGo shocked world by beating world Go champions. It collects playing data from pro matches for months and trains the model. AlphaZero collects all data from self-playing games, after 8 hours training it beats AlphaGo. Does it mean proper model is more important than "good" training data? Yes or no, it is impressive enough.
Gomoku is a board game which plays on Go board with 15*15 grid. Players take turn placing black and white pieces on an empty grid intersection. The first player who make 5 pieces in a row horizontally, vertically or diagonally wins the game. In this project, I try to reproduce partial idea of AlphaZero paper on Gomoku and learn corresponding techniques in the process.

2 Achievement

I build a working model for Gomoku, and fulfill partial goal of proposal.The best model achieves 46% testing accuracy and 55% "around 1 accuracy". From my personal learning perspective, i learned how to build NN by Pytorch, including GPU training and custom dataloader; some basic knowledge of convolutional NN; and various techniques while actual implementation.

3 Collection of play data

Influenced by self-training philosophy of Alpha Zero, my original plan is to collect play data by two Min-max AIs contest with each other, with tuned depth and random choices among best several moves. Intuitively randomness is guaranteed in this case.

However, after some practice, this idea seems not practical. In fact even with the "random setting" mentioned above, the evaluation function deeply restricted the randomness and variety of playing data. Due to the simplicity of evaluation function, self-playing moves tend to crowded with same colored pieces in a "dumb" way, Moreover, those games usually end very early because of the randomness. Lastly, generating those moves takes much longer than i expected.

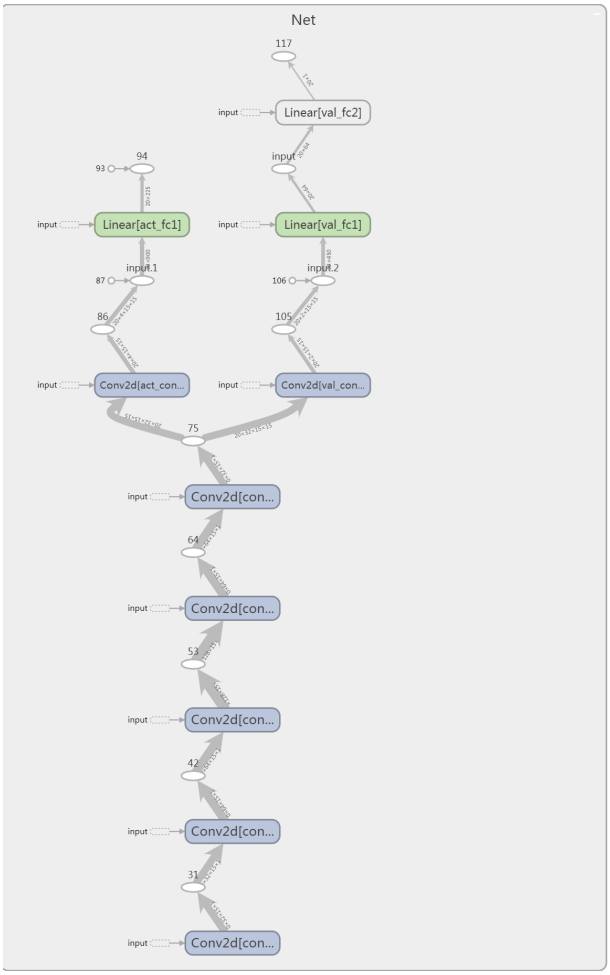
As a result, i collect sgf format play data from pro plays online, and use open-source reader class to cast them into array, then transform them into Pytorch tensor. The advantage is data is well distributed and does not emphasize on early stage of game.

4 Features and Labels

Feature is single board of a 15*15 matrix, which black pieces are represented as 2, white pieces are represented as 1. Label is also single 225 1-d tensor, on which there is only one non-zero value 2, representing next black move of the corresponding feature board. As it shows, the features and labels only represents black moves so we don't have to deal with extra features for representing who is current player (which is quite important in board game).
Features are transformed in form of (batch, 1, height, width) as required by Pytorch; labels are transformed into 0-d tensor, carries the argmax of next move. The neural network will predict one of 225 classes, each for 1 spot on the board.

5 Neural Network

Rather than building both policy and value network, i only make use of the policy network in this project. (if time allows i'll add value NN too) Following the paper and some open source implementations, i use convolutional NN with relu activation function between each layers. From my limited understanding, CNN has a good performance on identifying gomoku piece patterns, as it is very similar to discovering the relationship between nearby pixels in image classification.
There are 5 CNN layers with 3*3 filter, which is enough to capture most powerful pattern in Gomoku play, followed by one layer CNN with 1*1. After 1 full connect layer, it outputs argmax of 1d tensor with softmax function. The structure is shown in the graph, right branch abandoned.



6 Evaluation

Initially the play data consists of about 2300 games. I split 2000 of them to training data and 300 of them to testing data. Data is shuffled for each epoch.
The accuracy is based on if the NN predicts label, and not very precisely, if the prediction is very close to the label on the board. (1 spot) This measure can prove the model learns from nearby spots. After 20 epochs, training

accuracy grows from 13% to 34%, while testing accuracy grows from 9.5% to 18%.
In "around 1 accuracy", training grows from 27% to 45%, and testing grows from 25% to 29%.
After ecoch 13, both accuracy and "around 1 accuracy show sign of overfitting, Testing accuracies get stable as training keeps increasing.
Here is accuracies for 2000 games training set:

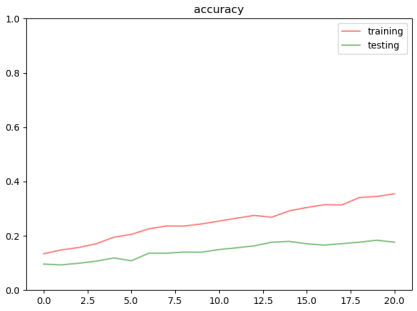


Figure 1: Accuracies after 20 epochs

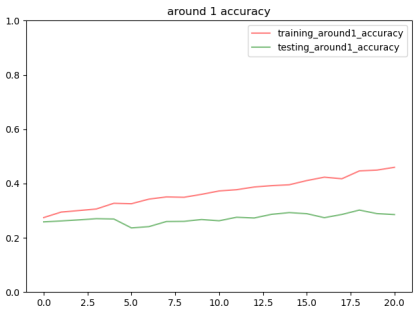


Figure 2: Around 1 accuracies after 20 epochs

After realizing the bottleneck could be the shortage of training data, i collected 5000 more games for the training set, end up with an obvious boost on testing accuracies. The model reaches 46% testing accuracies after 10 epochs. After epoch 10, the model shows sign of overfitting. Here is accuracies for 7000 games training set:

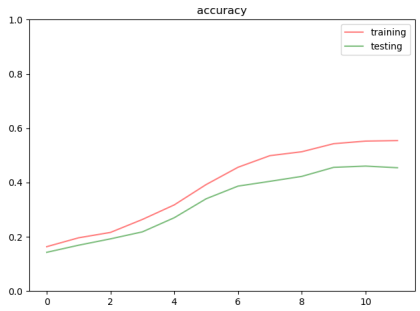


Figure 3: Accuracies after 11 epochs

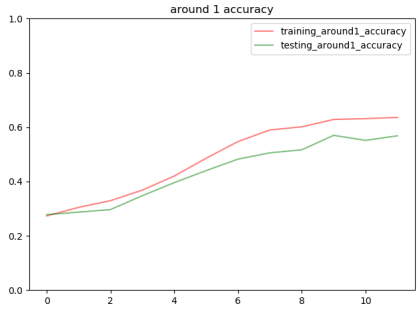


Figure 4: Around 1 accuracies after 11 epochs

For comparison, in a related paper [1] the group gets 42% testing accuracy by 10 layers CNN, with 66000 games training data, and TitanX GPU. Considering to the time and computing resources constraint, the result is surprisingly outstanding.

7 Future work

As we discussed in meeting, the labels collection from actual games are more complex and hard

to find pattern, which is not as good for practice as if we generate labels from evaluation function, since it's harder to build a working model base on my ability. In short term, i will feed evaluation function score along with the best prediction into the NN as labels and implement both regression and classification. Also if in need i can augment the board data by flip and rotation. In long term, i still hope to reproduce the original Alpha-zero implementation, This is definitely a good practice, from what i learned so far.
In the very end, thanks for the term and i learned a lot from you!

8 Reference

sgf file reader class, and partial training data is from here:
https://github.com/zhengtianqi94/AIGoBang
CNN structure is learned from the implementation here:
https://github.com/junxiaosong/AlphaZero_Gomoku

[1]Shao, K & Zhao,D et al. (2016). Move prediction in Gomoku using deep learning. 292-297. 10.1109/YAC.2016.7804906.

[2]Silver, D & Huang, A et al. (2016). Mastering the game of Go with deep neural networks and tree search. Nature. 529. 484-489. 10.1038/nature16961.

[3]Silver, D& Schrittwieser, J al.(2017). Mastering the game of Go without human knowledge. Nature. 550. 354-359. 10.1038/nature24270.