VRNN 2018 - Predicting legal chess moves

"Chess is the struggle against the error."

Mateusz Kiebala
University of Warsaw
mk359758@students.mimuw.edu.pl

Tomasz Knopik
University of Warsaw
tk359778@students.mimuw.edu.pl

Janusz Marcinkiewicz
University of Warsaw
jm360338@students.mimuw.edu.pl

Abstract

One of the recent problems that gained a lot of attention in Deep Learning field was playing Go and Chess at the professional level. The models for those problems are assigning the probability of the win for certain moves and positions where the moves are generated deterministically. In our work, we are checking if it is possible to learn a CNN network the rules of chess providing the only image of the board before and after the move. We test two different models architectures: single CNN model which processes two input images as one concatenated; and Siamese CNN model which takes two images separately, extracts features and combines them together predicting the result. Those two approaches achieve $\sim 90\%$ *on the training set and* $\sim 86\%$ *on the validation set.*

1. Introduction

Recent achievements in Deep Learning allowed researchers to beat top players in Go and Chess using Deep Learning models[3]. To learn such model authors have used techniques like Reinforcement Learning and Deep Convolutional Neural Networks to predict win percentages for given positions. Those models relay on deterministic move generation and known encoding of position.

In our work, we wanted to change the perspective of playing chess. We wanted our model to learn chess rules without prior definition of what they are and what is the purpose of the game. This is exactly what people, who did not see games like Chess or Go before, do when they see two other players playing those games – they try to understand the possibilities, rules and purpose of the game just by looking at how the players move.

We believe that this problem is super important. If the model can learn the rules of chess, maybe it also can learn the rules of the world. Long-standing question about single formula describing the world could be solved by CNN. It is still long way ahead of us but we believe we can make it and our work is the first step towards this achievement.

2. Related work

We have struggled to find any papers or work done in this area. All the papers we have found, are targeting learning model to evaluate positions but not how to play the game. In end-to-end chess playing paper [1] authors discuss a model to not only evaluate positions but also determining if the move is correct. We have used the key insights from the paper to improve our models' architectures. Another paper about predicting chess moves [2] provided us with a lot of useful information about using and not using different types of layers in our models.

3. Data

Chess is a deterministic game and the moves as well as positions can be easily generated and encoded.

Data used in our experiments was generated using python chess library[1] which allowed to save given position as images which then we converted into numpy arrays[2]. We have created a framework for generating random data with different outputs which were necessary for different types of models we have tested. The data was generated from random plays as well as positions and moves from book games[3]. We have combined these two sources into datasets on which our models were trained and tested. This way we wanted to prevent our model from overfitting on specific types of plays. Wrong moves were created randomly by putting pieces on random squares on currently processed

[1] https://github.com/niklasf/python-chess
[2] http://www.numpy.org
[3] https://www.ficsgames.org/download.html

chess board, and ensuring that such a move is not legal. The ratio of legal and illegal moves was around 50–50. What's worth noting is the fact that introducing additional illegal moves, which fallacy was due to checks or checkmates, had minor influence overall accuracy.

To train the models we have generated around 15 datasets with different random vs book games ratio each of them containing about $5'000$ moves. In total, we gained about $70'000$ moves to train on.

4. Methods

In our approach, we tried two different types of CNN models.

4.1. Single CNN model

One of the first models we trained is CNN model which consists of couple convolution layers followed by the fully connected layers[4]. The model, as an input, was fed with concatenated images: before and after a move. In this model, we wanted to check if convolution layers can extract more features having more information.

4.2. Siamese CNN model

Siamese CNN model consists of two identical CNN networks which output was then concatenated and passed to the fully connected layers[5]. This model was fed with two inputs: image before and after a move. Our understanding was that the model should independently extract features of each of the image and in the output have some kind of encoding which then was fed to fully connected layers.

4.3. Deterministic encoding

In our last method we tried to see what accuracy can we achieve if we pass encoding of the chess board to fully connected layer without the need of extracting features by convolution layers. This experiment was to see what is the bottleneck: too shallow representation of features extracted by convolution layers or fully connected layers. A term deterministic encoding means a chess board, encoded as $\{0, 1\}$ vector of length 768 – there are 12 different pieces (number of colors * number of pieces) each of them can occupy one of 64 positions.

4.4. Color vs grayscale

In the beginning, we used colorful images (3 channels). We suspected that the networks could have been learning specific features depending on color. Thus, we started generating images in grayscale (1 channel). Unfortunately, we did not observe any improvement and thus came to the conclusion that our convolution layers must have been generalizing quite well.

[4]Single model architecture: `http://students.mimuw.edu.pl/~jm360338/vrnn2018/model_single.png`

[5]Dual model architecture: `http://students.mimuw.edu.pl/~jm360338/vrnn2018/model_dual.png`

5. Experiments

In the experiments, we have tried different approaches for different types of models. In both types of models as well as deterministic encoding we achieved $\sim 86\%$ on a validation set.

What may be interesting is that the deterministic encoding had similar accuracy as the other models what suggest that convolution layers did a great job in extracting features and the bottleneck in the accuracy was caused by fully connected layers. Worth noting is the fact, that thanks to the compact encoding representation, we were able to train a network on a much bigger dataset of roughly $200'000$ moves. However, we observed that it did not have a significant impact on results, as final accuracy reached $\sim 87\%$. With this information, we tried to implement a different set of fully connected layers but without further success. We decided to not go further with this part of a project, as this is out of the course's scope.

5.1. Hardware

We trained our models on single 1080Ti GPU. Each of the models was trained on ~ 5000 epochs with 64 batch size. Single dataset was trained for 10 epochs, after that new dataset was loaded and it was trained for 10 epochs as well, then whole process repeated. As we aforementioned it before, we have generated 15 different datasets – each of them contained new images and some of them had different random/book games ratio. Therefore, each of the datasets was trained around 30 times.

5.2. Method comparison

Method	Dataset size	No. epochs	Accuracy
Single	$15'000$	$10'000$	85.2%
Siamese	$70'000$	$4'000$	86.2%
Deterministic encoding	$200'000$	$10'000$	87%

6. Conclusions

At the very beginning of the project, our goal was to solve a problem of pattern recognition and logical reasoning based on rules of chess. We did not accomplish any significant results taking into consideration solely the given task. However, our experiments turned out to be successful in terms of image recognition problem.

Our project contributed to show that logical reasoning is a hard problem even if it is limited to a specific use case.

It shows that well-crafted convolution neutral network successfully extracts desired features from images, even when a loss function describes an abstract concept, like the legality of chess move, rather than a simple idea like a class of an object on an image. The image recognition overhead is relatively small ($\sim$ 1-2 percentage points), compared to the perfect representation of deterministic encoding.

There are plenty of opportunities for further work. They include, but are not limited to:

- testing proposed solutions on better hardware and much larger datasets, as we provide a convenient way to generate an almost unlimited dataset,
- evaluating our CNN solution with better-designed neutral network for logical reasoning part.

References

[1] E. David, N. S. Netanyahu, and L. Wolf. Deepchess: End-to-end deep neural network for automatic learning in chess. *CoRR*, abs/1711.09667, 2017.

[2] B. Oshri. Predicting moves in chess using convolutional neural networks. 2015.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, Oct. 2017.