Determine Chess Game State from an Image

Project Exploration Conducted by: Ryan Lipski

ITCS 5156: Applied Machine Learning

28 February 2022

Based on an Original Paper: Determine Chess Game State from an Image

Authored by: Georg Wölfein and Ognjen Arandjelović

Originally Published: 2 June 2021

Published In: Journal of Imaging 7, no. 6: 94

1 Introduction

1.1 Problem Statement

The main problem I hope to solve is to see if a model can be created to identify chess pieces regardless of the style (shape and color). Solving this problem would be useful in automating the recording of the final state for chess tournaments. The process of manually recording the game state does not take long, however speeding this up would obviously be easier.

1.2 Motivation and Challenges

While there have been other models that have developed to identify the game state of from an image, most are trained on specific chess sets or require monitoring the progress of the game from the beginning. Therefore, it would be helpful to build a model that can take variations in chess sets into account when recording the game state. If there is some success in this, we can possibly apply this to other classification problems where an object can appear in many shapes/colors.

Building this kind of model has some challenges, even without introducing new chess sets. The biggest challenge is the fact that chess sets are 3-dimensional, which means that all game pieces may not be completely visible depending on the angle that the picture of the board is taken at. Since the main problem I am trying to solve is developing a model to work on multiple styles of chess sets, I will introduce some basic constraints to reduce the complexity of this problem. The constraints be summed up as using only digital, 2-dimmensional chessboards that are always oriented in the same way. These constraints will help prove that different styles of chess pieces can still be identified as the pieces they represent.

1.3 My Approach

My approach will follow closely with the approach that the original paper took. Given the constraints that I have placed on the problem, certain portions of the original model will not have to be implemented. The relevant portion includes an occupancy classifier, which determines whether a given square is occupied or not, and a piece classifier, which determines what piece is currently occupying the given square [1]. To evaluate the model, I will be looking at whether the entire chessboard is accurately recorded, since that is the main purpose of this model.

2 Related Work

2.1 First Paper

The first paper that I looked at was written in 2016 and sought to track the location of the game pieces as the game is played [2]. This method uses a bird’s eye view to view the board and track the pieces [2]. Using the squares on the board, the model can properly crop the image so that the chessboard takes up the entire image [2]. From there, the model determines which side is white and black using the average color of the king square from each side and then orients the board so white is at the bottom of the image [2].

Before the game proceeds, the model takes a reference image of the initial state of the board and the average color is taken from the center of each of the occupied squares [2]. After a move is made, the model compares the new occupied squares with the old to see which piece was the one to move [2]. Due to the fact that the model knows the initial state of the board, it makes it easier to identify the pieces as the game progresses.

The pros of this method is that, since the tracking is present from the very beginning of the game and the average color is taken then as well, it will likely do very well on differently shaped chess pieces. Another pro of this model is that it can document each move of the game rather than just the final state. This is useful to document how the game progresses so that players can easily recreate how the game was played out

The cons of this method is that the camera must be present at the very beginning of the game. Without knowing the average color of each of the game pieces, it makes is very hard for the model to determine which pieces are which. Additionally, due to the fact that the model compares the color of the two king squares in order to see which is white and which is black, it is likely not going to perform well on oddly colored chess pieces.

This approach is not very relevant to my approach. It was, however, interesting to see that there are many ways to solve the same problem.

2.2 Second Paper

The second paper that I looked at was written in 2020 and sought to identify chess pieces with the aid of probability [3]. Similar to the previous method, the model is able to identify and correctly orient the board in the image using the squares on the board [3]. The key difference between this method and the previous one is that rather than identify the pieces by tracking their position throughout the game, they use a chess engine to determine the likelihood of the set of pieces being set up in a given configuration [3]. The researchers found that this addition was useful since many of the game pieces can look very similar from above [3].

The pros of this method is that, the addition of the chess engine allows for the reduction of classifications that result in illegal or unlikely configurations for the game pieces. This method also does not seem to require for observe the game from the beginning in order to identify the pieces. Additionally, this method would likely do well with different chess sets, regardless of the shape and color.

The cons of this method is that, while the chess engine helps to reduce misclassifications. However, this does not completely eliminate them. In the paper, there was not a 100% training, validation, or testing accuracy [3].

This method would be an interesting approach to explore. However due to the dataset that I chose, there were cases where some of the data contained illegal positions for some of the pieces. This is due to the fact that each piece was placed based on a probability instead of as a result of actual gameplay [4].

3 Chosen Method

3.1 Paper

The paper that I chose to base this project off of was written in 2021 and sought to identify chess pieces from an angled side view, rather than a bird's eye view [1]. Similar to previous methods, this model also identified the board by using the squares on the chessboard [1]. From this point, the model is broken up into two different classifiers: the occupancy and piece classifiers [1].

The occupancy classifier classifies each of the squares as occupied or unoccupied [1]. The model that these researchers found to work best was the pretrained residual network-18 [1]. This model was able to give them a validation accuracy of 99.96%, however all other models that they attempted to use performed very similarly [1].

The piece classifier classifies each of the squares that were determined to be occupied as one of the 12 different pieces [1]. The model that they found to give them the best performance was the InceptionV3 model [1]. Similar to the occupancy classifier, they had great success with these models; they were able to achieve 100% accuracy on the validation set [1].

The pros of this method is that it is helpful to be able to take a picture of the board from different angles, rather than just from above. Additionally, there is not a requirement for this model to observe the game from start to finish.

The cons of this method is that is unlikely to perform as well on a different chess set due to the fact that the model was trained on a specific style. Additionally, depending on the angle that the picture of the board is taken from, there is a chance that some of the game pieces will be obscured by others, which can make classification harder to accomplish.

3.2 Adaptation of Method

Given that I introduced some constraints onto this problem, only a portion of the models that were implemented in this method are required. Specifically, only the occupancy and piece classifiers are required. This is because the input data that I am using is always in the same orientation, which means that the squares will always be in the same area.

The occupancy classifier will be able to be implemented in a very similar way that the original researchers implemented theirs. However, due to the limitation of the input size of the InceptionV3, I will have to choose another model that performed well on the piece classifier. Since I will already be using ResNet for the occupancy classifier, I chose to utilize the ResNet for the piece classifier as well. Additionally, while I will still have both of these classfiers, I will be classifying the pieces to include an empty square. This is due to the fact that I would be able to train both these classifiers from the same set of data without separating them.

4 Experiments

4.1 Setup

Due to the fact that I would be training a CNN, I decided to do the modeling on Google Colab. The dataset that I used was a large dataset from a user on Kaggle [4]. This dataset, originally contained 100000 images of chessboards, split 80:20. However, since the classifiers were going to be trained on individual squares, I was going to have to break the images down to 64 individual images. If I were to break down all of the training data into these squares, there would be 5120000 images to train from. However, due to computational time limits from Google Colab, I would be unable to use all of these images. Therefore I chose to 2000 random images from the training set and broke them down. This resulted in 128000 images to train from, which I felt would be far more than enough. Additionally, I chose 500 random chess boards from the test set and broke those down. This is because, rather than break each image down during testing, it was easier to just maintain the order of the images and then compare the results as groups of 64 (a full chessboard).

Using methods from the paper and from class, I set up the datablock/dataloaders to properly import the training and test sets. This includes two functions that is able to identify the class of the square based on the name of the file. The occupancy classifier used a batch size of 256 while the piece classifier used 128. The piece classifier batch size matched what was described in the paper, however the occupancy classifier was set to 256. This is due to an error on my part; however, it did not negatively affect the results. From there, I set up two ResNet-18 models, each with the final layer modified to include the proper number of classes (2 for occupancy and 13 for the piece). These models used Cross Entropy as the loss function and Adam as the optimization function since that is what was described in the paper. While the paper describes training each of these models under 3 epochs (1 time with frozen layers and a learning rate of .001, and the rest with unfrozen layers and a learning rate of .0001). I chose to only do the first epoch, which I found did not negatively affect the results. The training loss for the occupancy classifier was .008777 and .016619 for the piece classifier. Since the models were now trained, I ran the test set through each of the classifiers.

The testing of these models involved 500 chessboard configurations. Since it was easier to simply break the chessboards into individual squares myself and keep the order of the images intact, I choose to do that. Additionally, due to this fact, it was difficult to keep track of which squares were misclassified by the occupancy classifier (if any). Fortunately, the occupancy classifier did not misclassify any of them. From there the squares were passed through the piece classifier and the results were saved.

4.2 Results

The testing of these models involved 500 chessboard configurations. As stated before, the testing was done using all of the squares at once, while maintaining the original order so each chessboard was still grouped together.

The occupancy classifier was able to properly classify 100% of the squares that it analyzed. The piece classifier had a total of 12 chessboard configurations incorrectly identified. However, there were a total of 14 errors (2 boards had 2 errors). Working under the assumption that we want the entire board to be correct, the entire model was able to correctly identify 97.6% of the chessboards that it looked at. If we are just looking at individual squares, then the accuracy was 99.96%.

4.3 Analysis

These results are comparable to the results that the original paper found. This generally makes sense since ResNet has been used to classify similar problems, so it makes sense that it can be applied to this one. The increase in performance for the occupancy classifier is likely due to the fact that the board is always the same and all squares are always visible.

The slight decrease in performance in the piece classifier is likely due to a couple reasons. The biggest being the fact that only one epoch was done for this model. The described method had a total of three in order to achieve the accuracy they did. The other possible reason is the training set was a random selection of all the different styles. This means that the there might have been a style that was not present during training but was present during the testing. If this was the case, then only 14 total errors is a good sign that a model can be developed to identify different (and potentially unseen) styles of chess pieces.

5 Conclusion

Overall, the results of this project are quite positive. Each of these models show that you can train a model to identify different game pieces regardless of the style, with fairly high accuracy. It was interesting seeing how the problem of identifying the game state of a chess game can be attempted to be solved in so many ways. I still believe that that paper I based this project on, is the most promising for solving this problem. I hope that this paper shows that that model can be expanded upon to include additional chess sets in the hopes to be able to utilize that model regardless of the chess set.

From the perceptive of a student this project was interesting to me. I was able to apply some of the methods and techniques that I learned within this class to understand and utilize the methods that were described in the paper. Seeing this process from close to the start and finish was very valuable to me a student. Despite challenges with debugging of input to these ResNet models, I gained additional practice in reading poorly explained documentation.

6 My Contributions

My contribution to this project is the application of the types of models that were used in a previous work were applied to a new dataset. This involved me preparing the dataset to be used within the context of those models.

The portion of my work that came from existing work was the dataset and the base models themselves. The dataset came from a user on Kaggle [4]. While the choice on the ideal models to use for these two classifiers came from paper that was the basis for this project [1].

7 References

[1] Czyzewski, Maciej A., et al. “Chessboard and Chess Piece Recognition with the Support of Neural Networks.” *Cornell University, ArXiv,* 13 Aug. 2017.

[2] Koray, Can, and Emre Sumer. “A Computer Vision System For Chess Game Tracking.” *21st Computer Vision Workshop,* 3 Feb. 2016.

[3] Wölflein, Georg, and Ognjen Arandjelović. “Determining Chess Game State from an Image.” *Journal of Imaging,* vol. 7, no. 6, 2 June 2021, p. 94., https://doi.org/10.3390/jimaging7060094.

[4] Koryakin Pavel. 2019 Feb. “Chess Positions”, Version 1. Retrieved 14 February 2022, https://www.kaggle.com/koryakinp/chess-positions

8 Sharing Agreement

Please do not distribute my work during future semesters of this course.