2D Chess Piece Classification using CNN

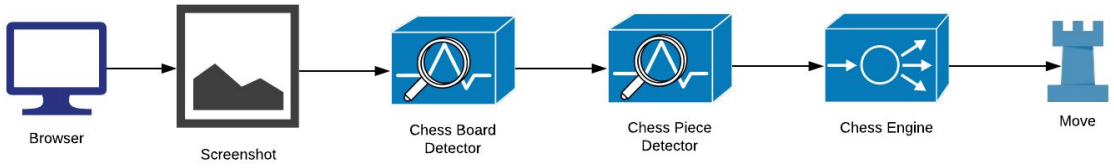
Rajtilak Indrajit
December 3, 2018

Proposal

Domain Background

Chess is a two-player strategy game based on an ancient Indian board game called Chaturanga. In the past twenty years, advancements in computing has propelled the efficacy of chess engines at game analysis.

The focus of this capstone is to take the first step to enable real-time engine analysis of online chess games through a Chess Bot. The central component involves developing a robust classifier for chess pieces. Future work will involve using this classifier to convert online chess games to the standard FEN notation, which in turn can be used for engine analysis and move suggestion. Enabling this at high level involves identifying the chess board, classifying the chess pieces, interpreting the chess position, and feeding it to an engine for analysis. This capstone will focus on chess piece classification.



There is existing work in this domain. TensorFlow Chessbot by Sameer Asari uses CNNs to perform this task on 2D chess boards and achieves fairly robust performance. Chess Piece Recognition by Anoshan Indreswaran also takes a similar approach. My goal is to build on top of these projects.

Chess is a game that I am particularly fond of and recently been following closely. The world chess championship that was held in November 2018, is still fresh in my memory. I also play Chess online for fun on lichess.org.

Problem Statement

The core problem to be solved is the classification of chess pieces dataset into the six corresponding classes - King, Queen, Rook, Bishop, Knight, Pawn, and Blank.

Since chess pieces come in all shapes and sizes, a traditional computer vision approach to classification will fall short. Hence, the hard problem is to develop a generic and robust model that can adapt to variations in the a) shape, b) size, c) color, and d) design of the chess pieces.

The primary success metric is classification accuracy. However, it is important that the accuracy is measured on real-world datasets, using non-uniform chess pieces.

Datasets and Inputs

Internet searches revealed that there is not an existing chess image dataset that can be used for this classification task. There are some additional criteria for our dataset, we require a dataset that is non-uniform and representative of the variability of chess pieces in the real world.

Hence, I decided to build a dataset of chess piece images from real-world images of chess pieces. Using various chess piece and board designs from lichess.org, chess.com, chess24.com as input, a dataset of 1k images were manually generated. The dataset has images that vary in background, texture, shape, color, and design. This variability is a close approximation to online chess boards that the model is meant to cope with. Additional variability in zoom, shear, and rotation is introduced during training and testing. The dataset will be used for both training and validation of the model. The test set consists of different chess board design, collected from

Solution Statement

The solution uses Deep Learning and Convolutional Neural Networks to classify the chess piece images. The model trained on a dataset with ~200 images per class. The model is designed to be robust and adapt to the variability in chess board and piece design online. The primary metric is classification accuracy.

Benchmark Model

Since the solution hinges on the eigenvector approach, the most classic benchmark model would be a Multi-layer Perceptron model that leverages Principal Component Analysis. The primary metric is classification accuracy.

Evaluation Metrics

The primary evaluation metric used will be classification accuracy. However, since a board consists of 64 squares, and we need to classify all of them accurately, a secondary metric could be board accuracy, that combines all 64 squares are predicted accurately.

Piece-wise Accuracy = # true piece predictions / total

Board Accuracy = # true board positions / total boards

The goal is to achieve a piece-wise accuracy of 0.9 or greater, and a board accuracy of 0.7 or higher.

Project Design

Within the realm of chess piece classification, there are a number of approaches that I considered. Some example questions include detecting individual pieces vs entire positions, 2D vs 3D board and piece support, detecting pieces vs detecting pieces with color, using images from the web vs

generating images from chess boards online. This was an iterative process where I spent a few tens of hours trying one approach, learning, and iterating.

Ultimately, the design I landed on is the most simplistic, but also the most effective. The process steps include creating the dataset by sourcing and manually classifying the training and validation set into the corresponding image classes. The images will then be converted to grayscale and scaled to desensitize our detector to color and size variations. These images are then used to train a CNN that uses 5 convolution layers and nearly 100k trainable parameters. The resulting model will then be run on our test set to measure classification accuracy.

Next Steps

Future work would include building other parts of the overall Chess Bot. This includes a Browser extension that supports screen capture, a chess board detector, an image splicer to convert the board to individual images, the neural network classifier (which we developed above), a piece color detector, an FEN generator that combines the detected pieces, and finally, a chess engine that will predict the next move.

Future Work

Future work can focus on a number of areas of improvement. For instance, we could start supporting 3d chess pieces instead of limiting to 2d only. Likewise, we could also support more skew and transformation of images. Similarly, the model could be extended not only to predict the piece but also the color.