Capstone Proposal

Convolution Neural Network Chess Board Recognition

Daniel Cuomo

9/26/2018

1. Domain Background

Supervised learning, specifically deep learning is a growing field in machine learning that is being used to tackle a range of challenging problems. One area in specific that deep learning, specifically neural networks, is expanding to is computer vision. An important area of computer vision is object recognition, which is important for applications such as autonomous cars being able to identify road sides, facial recognition for social medial applications or estimating orientation of an object for an assembly line.

Convolutional neural networks have helped revolutionized this field, replacing previous computer vision methods with more advanced techniques based on supervised learning. (Computer Vision Wikipedia, n.d.) Some challenges due present themselves as now a training requirement is added to image processing, but for a large portion of applications that data is present or can be obtained.

2. Problem Statement

Modern chess players to improve spend countless hours revisiting positions from previous games, largely in part from memory or written notation. However, depending on time formats recording of all moves in a game is either not necessary or not easy. An automated method for capturing game moves or game state could help immensely. Players could then obtain a log file afterwards that transcribed game moves and board position at each state, allowing them to review and study positions later without fear of transcription errors. There exist similar tools, but a majority require starting from the initial board state (Meyer, n.d.), not allowing for chess variants like Chess960 or practicing tactics problem or end game scenarios from a known position. This is a classification problem in nature, as it is identifying the pieces. Chess piece identification has been explored in an academic setting using classical computer vision techniques and CNNs. (Xie, Tang, & Hoff, n.d.) (Danner & Kafafy, n.d.)

3. Dataset & Input

The dataset for this problem will be manually collected via a top mounted camera at a fixed position. The pieces will be positioned randomly on the board, always a full set and moved to ensure each type of piece is at least in each square once. There will be 1800 black and white individual square images in the dataset containing either one of the pieces types or an empty square all the same size 135x135. The split of the dataset will be 60% training, 20% validation and 20% testing, this will be done by piece type to ensure the proportions are consistent. Most tournaments across the US at the lower levels use a standard chess board shown in Figure 1, this will be the board and pieces used for data collection.

Figure 1 Standard Tournament Chess Set

The collected images will then be divided into 64 individual images each representing a square on the 8x8 chess board. These images will then be identified either as a dark or light empty square, or a dark or light pawn, knight, bishop, rook, queen or king. These inputs will then be used to help train the classifier to determine if a piece is present in a square and what piece is present.

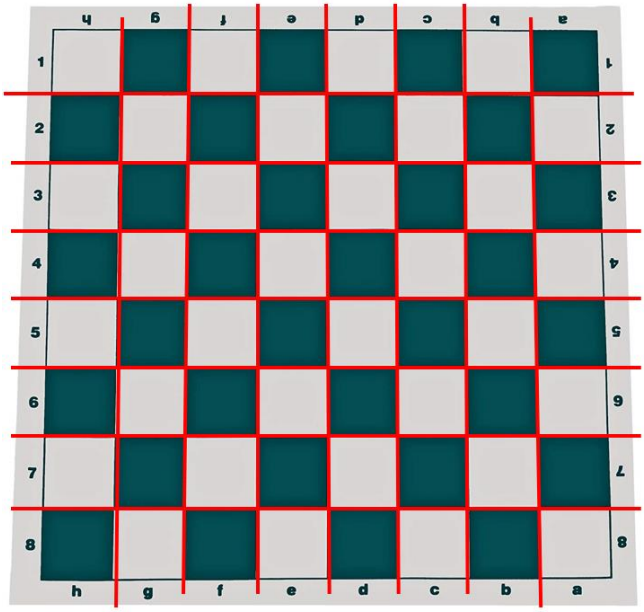

Figure 2 Chess Board Input

4. Solution Statement

To solve this problem a CNN will be developed and train specifically with the goal of identifying the state of a square on a chess board. When given a full board it will examine each square individually and return a digital representation of the board. Both a custom CNN architecture as well as a transfer learning approach will be evaluated for performance.

5. Benchmark Model

For this problem, the benchmark will be a simple CNN agent, this will likely perform very poorly, given the amount of possibilities present for each square, so the trained CNN should significantly outperform this. The simple agent will consist of three convolution layers and final dense layer for the classes.

6. Evaluation Metrics

There will be two key metrics used in evaluation of the model. The first will be Keras categorical accuracy, this checks that the max predicted value probability lines up with categorical label provided with the agent across all classes. (Keras metrics, n.d.) At its core this is an accuracy problem, as failure to identify a piece correctly would invalidate the effort.

Another key evaluation metric which will be used during training which will be categorical cross entropy. This is similar to log loss, the loss increases as the predicted probability diverges from the truth label. This will help to prevent the model from favoring the classes that have more training points present, specifically blanks quares.

7. Project Design

Software: Python 3 w/ Numpy, MatplotLib, OpenCV and Keras

Below shown in Figure 3 is the general plan for the program flow. There will be a class responsible for the general image preprocessing. It will dissect the chess board into the 64 individual pieces and store them in a dictionary with a default value. After the preprocessing is completed it will be passed through the CNN agent, the CNN agent will then assign a label to each by updating the default value. Then the output generator will be responsible for generating a digital interpretation of the chess board and generating a corresponding output file.

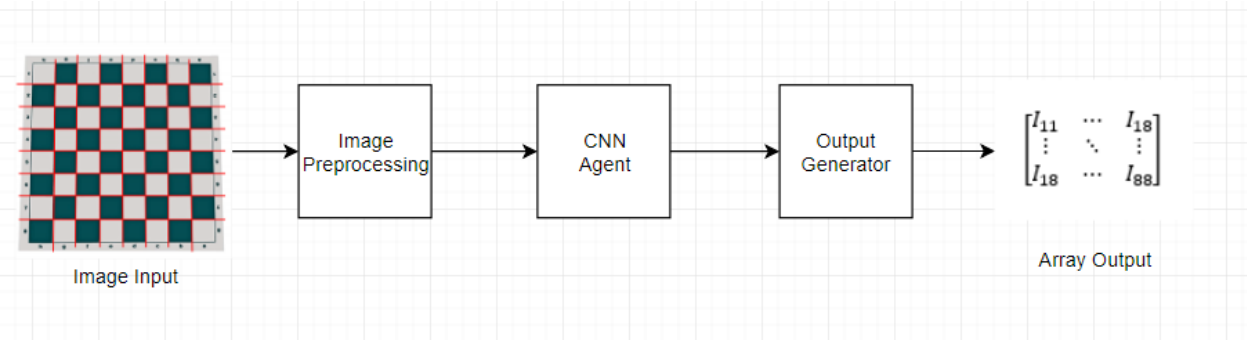

Figure 3 Prediction Program Flow

The program has two main modes, one is for training and the other is for predicting. Each will take advantage of three different objects that will have key functions described below:

1. Image Preprocessing Object:
    a. Training Mode: Will dissect chess board into 64 individual images, will then display image and prompt for user input for what piece. Individual image will then be saved to specific folder corresponding to appropriate piece.

    b. Prediction Mode: Will dissect chess board into 64 individual images, then store each in dictionary with a default value. Output will be passed to CNN agent.
2. CNN Agent:
    a. Training Mode: Object will receive a set of training, test and validation images containing individual squares. Object Will use keras to train CNN, two different models will be checked, custom CNN as well as a transfer learning approach. Best weights will be stored to a file to be used in prediction Mode:
    b. Prediction Mode: Will use best weights file from training mode and take in individual square images from a dissected board to predict piece in square.
3. Output Generator:
    a. Training Mode: Will output the accuracy and other key metrics for the CNN.
    b. Prediction Mode: Will output a digital representation of the 8x8 chess board passed through the predictor.
        i. Note: due to the differences in these two features, these may be separate objects for each purpose.

The image processing mode will be broken into a few steps. The first portion will be to take advantage of the OpenCV built in algorithms to find the inner corners of the chessboard and then extrapolate out the edges. After locating the outer edges, a perspective transform will be performed to ensure it is a square image only containing the chessboard. Then it will be sliced into 64 black and white images of 135x135 pixels. These will be passed to the neural network for training or to make predictions.

The convolution neural network will build off the baseline mentioned in section 5. The key thing will be varying the number of filters and size of them in the convolution layers, as well as the number of layers. Additionally, the effects of batch normalization, dropout, and different pooling layers will be assessed. The last big factor will be addressed the benefits of image augmentation to make up for the smaller size of the dataset.

8. References

Computer Vision Wikipedia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Computer_vision#Recognition

Keras metrics. (n.d.). Retrieved from Keras Documentation: https://keras.io/metrics/

Meyer, J. (n.d.). *Raspberry Turk*. Retrieved from Raspberry Turk: http://www.raspberryturk.com/