ChessLogger: Chess Piece and Game State Recognition

Jordan James
University of Texas at Arlington
701 S Nedderman Dr, Arlington, TX
jaj9608@mavs.uta.edu

Abstract

The task of converting a physical chess game to a representation which can be used online is a significant challenge in the realm of computer vision. As major chess tournaments are played over the board, the solution to this problem is of interest to many members of the chess community. A portable system capable of logging a chess match would allow for automated broadcasting, easy analysis of games, and even over the board online play. In order to log fast paced game modes, such as bullet and blitz, such a system requires low latency. Current work has shown promising results but lacks the fast response time required to log game modes with lower time limits. This paper proposes a system to decrease the latency of game state recognition by splitting the task into a system of parallel processes, one process to detect and recognize the chess pieces in a given image and another process to locate the board and extract the squares from the image. To allow for portability, inference of the chess piece recognition has been tested on the NVIDIA Jetson Nano 2GB. The implemented system achieves a mean average precision of 99.4% for chess piece recognition at 1 frame per second and locates the position of the pieces on the board within one square.

1. Introduction

Chess is an extremely popular game played by many people across the world. The game is traditionally and primarily played in person over the board and many small tournaments as well as most large tournaments are held in person. As the games are all over the board at these events, players have to record their games as they play in order to have a log which they can take home and analyze to improve their play. Similarly, broadcasters at these events must manually follow games that are broadcast live. While traditional games of chess have long time limits and allow for players and broadcasters to manually record moves, this is not the case for game modes with decreased time limits. Game modes such as bullet, blitz, and rapid have been rising in popularity and have even seen play at tournaments. In games such as these, players are making moves within seconds which leaves little to no time for logging and makes the task of manually following a game much more challenging for a broadcaster. As such, a system to log chess games that can easily be transported to and from tournaments is highly desired.

This paper details the implemented system which splits the task of chess piece recognition and chess board detection and fuses the results together to provide a virtual representation of the game state of the board. Recognizing the chess pieces directly from the image removes the need for the possible 32 recognition tasks and allows for less work to be done by the recognition process. Along with the reduced work for piece recognition, the process can be completed at the same time as the chess board detector and square extractor as it no longer requires the output to find pieces.

The outline of the paper is as follows. Section 2 gives a literature review and brief overview of related works. A description of the problem is given in greater detail in section 3. In section 4, the dataset used to train and test the chess piece recognition network is discussed and briefly outlined. The methods and algorithms used for the parallel processes responsible for digitizing a chess board is discussed in greater detail in section 5. The results for chess piece recognition, the board detector, and the fused system are covered in section 6. Section 7 discusses future work to be done on the problem followed by conclusions in section 8.

2. Literature Review

The problem of digitizing a chess board is a hot topic in the field of computer vision. Most current solutions to the problem involve a serial approach of detecting the board, extracting the squares, and then classifying the pieces on each of the extracted squares. The process of finding the board and extracting the squares tends to be the most important step in the process as the piece recognition and occupancy determination directly relies on the correctness of the output of the process. As such, many different meth-

ods of extracting the squares have been researched. One such approach that stands out from the rest is the method proposed by by Georg Wölflein and Ognjen Arandjelović which achieves a per square accuracy of 99.83% [1]. The method involves using a RANSAC based approach to compute a projective transform of the board onto a 8x8 grid. This approach, while accurate, has some latency and is required to be done serially making it less feasible for a live log. Further work has been done to optimize this algorithm to allow for the projective transform and occupancy classification to be done in under a second for use in live applications by Quintana, García, and Matías [2]. Another method which performs very well is the iterative approach presented by Czyzewski, Laskowski, and Wasik which detects straight lines, finds lattice points, and positions the chess board using a heat map based approach multiple times [4]. This approach however must be done in serial similar to the previous methods.

The second major process in the system is the chess piece recognition task. The state-of-the-art research for this problem involves the use of neural networks to predict the chess piece in a given image. Many of the approaches perform very well and multiple architectures have been researched for this task. The most notable architecture used for piece recognition is InceptionV3 which achieved 100% validation accuracy in the work done by Wölflein and Arandjelović [1]. Another model that achieves impressive results is VGG16 which was implemented by Afonso de Sá Delgado, and Campello to achieve 97% accuracy for chess piece recognition [3]. One such method that stands out is the use of support vector machines (SVM) by Matija as it is one of the few approaches that do not use neural networks for recognition [6]. Although the results of the SVM are good, it does not compare to the state-of-the-art methods of using deep neural networks. Based on the current research on the task of chess piece recognition, it is clear that learning based approaches and particularly neural networks are a strong solution for achieving accurate results.

3. Problem Description

Over the board chess games require manual logging by players and broadcasters which becomes a challenge for game modes with low time limits such as bullet and blitz. A system which can log an over the board match and provide a live feed for broadcasting is highly desired amongst tournament organizers and players. For a system to be capable of logging any chess game, it must have low latency in order to keep up with the moves played in game modes such as bullet and blitz. The task of digitizing a chess board is composed of multiple processes which are typically performed serially which results in a latency that is too large for fast paced game modes. To digitize a chess board, typically the board is detected in an image and each square is extracted from the image. These squares are then analyzed to determine occupancy and the squares determined to have pieces on them are then further analyzed to determine which piece is on the square [2]. Assuming a perfect occupancy detector, a worst case of 32 pieces still require analysis which leads to slow performance. This paper explores an architecture for game state recognition which allows for parallelization of the processes in order to reduce the latency required to digitize a chess board and tests the inference on the NVIDIA Jetson Nano to allow for portability and practical use of the system.

4. Dataset of Rendered Chess Game State Images

The dataset used for experiments and training of the chess piece recognition network in this paper is the Dataset of Rendered Chess Game State Images. This dataset consists of 4,888 images of rendered chess game states taken from tournament games played by Magnus Carlsen. For this paper, only the first 500 training images and the first 25 validation images are used. The images are synthetic such that they are not of a real chess board but are instead rendered based on an input camera matrix defining both intrinsic and extrinsic properties as well as the lighting scene for the image. The chess board and pieces are all wooden with black and white pieces consisting of different shades of brown to make the recognition task more challenging. To add to the challenge the table in which the board is placed upon is also wooden and of similar color to the board. Each image has a corresponding json file that specifies the parameters of the lighting and camera matrix used to generate the image as well as ground truth labels and bounding box locations for each piece on the board.

5. System Design and Algorithms

The task of digitizing a chess game requires multiple processes working together in unison, particularly board detection, square extraction, and chess piece recognition [1][2]. Typically these processes are performed in serial however in order to reduce latency, this paper presents a parellelization of the processes. The implemented system performs board detection and square extraction in parallel with the process of chess piece recognition and adds a final combination process to output the digitized results. The system is implemented entirely in python 3 using standard packages such as numpy, tensorflow, pytorch, and openCV.

5.1. Board Detection and Square Extraction

The detection of the board as well as the individual squares of the board are both dependent upon the corner points of the squares on the board. As the system is intended to operate in a full game of chess, pieces will be

present on the board at the time of detection and thus the process of finding corner points needs to be invariant to the pieces and only return corners of the squares. Due to this stipulation, classical methods of finding corner points in an image, such as the Harris Corner Detector, are not suitable. Instead a serial chain of processes are used to extract the desired corner points from the image.

The initial process in the chain is edge detection using the Canny Edge Detector. Since the table that the board rests on is wooden and has a large amount of texture, the image is first smoothed using a kernel of size 4x5 to avoid extra edges found from the table. The image is then converted to grayscale and passed to the Canny Edge Detector which uses an aperture size of 5 for the Sobel operator and a first and second hysteresis threshold of 1150 and 1550 respectively. Each of these values were determined experimentally by passing training and validation images and visualizing the edges returned.

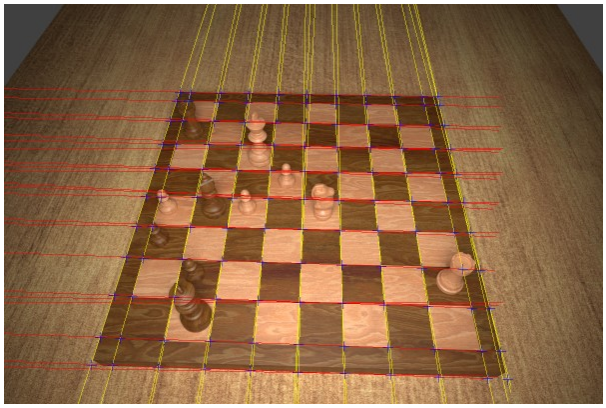

Figure 1. Segmented Hough lines and filtered corner points

Once the edges have been obtained from the image, the resulting binary image is then used to get the Hough lines using the Hough Transform. The accumulator threshold for the accumulator was chosen to be 65 experimentally as this allowed for all the required lines for each row and column of the board to be found for a large portion of the dataset images. As the threshold is set low, extra lines which are not required are also found which are then filtered out by angle. Only lines near 90 degrees and 180 degrees are kept as these are the lines which correspond to the rows and columns of the board. To avoid extra vertical and horizontal lines the rho and theta values of lines under consideration are compared to lines already chosen to be kept such that the extra similar lines can be discarded. Finally the filtered lines are then segmented into horizontal and vertical lines and the intersection points between the two segmented groups are found to give the corner points of the squares. The resulting corner points are then filtered using non-maximal suppression to remove extra points near other points. The process can be visualized in figure 1 which shows the segmented Hough lines in yellow and red and the filtered corner points as blue crosses.

The squares of the board can be extracted and digitized using the filtered corner points from the previous process. A square can be represented by the top left corner point and the bottom right corner point. To determine that two corner points form a square, the angle between the two can be checked as it should be approximately 45 degrees clockwise from the origin. The points can be iterated through taking each starting point as the top left corner point and the second point can then be searched for. As there can be many squares ranging from the top left of the board all the way to the bottom right, the points are sorted in a vector by row and column such that they appear in order from left to right and top to bottom and the first point found with a 45 degree angle is the smallest square and thus the intended one. The range of search can be limited as there is no reason to find squares after the first and thus processing time can be reduced. The found squares are stored as pairs of corner points in an 8x8 matrix to later be used with the result of the piece detector.

5.2. Chess Piece Detection and Recognition

The chess piece detection process is responsible for recognizing the pieces on the board in a given image as well as locating the pieces in the image. The architecture chosen for the task is You Only Look Once (YOLO) version 3 as it provides real-time object detection outputting the confidence of predicted labels as well as the bounding box locations of predictions [7]. YOLO uses a single neural network that solves the object detection problem as a regression problem and is able to give predictions in a single evaluation making it fast. As YOLO is only a single network, it is computationally feasible to deploy in a mobile setting making it a strong choice for the task of chess piece detection for the proposed system. An example of the output of the chess piece detection can be seen in figure 2.

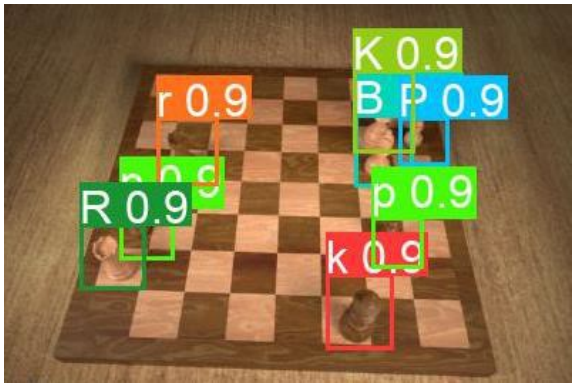

Figure 2. Output of YOLOv3 chess piece detection

5.3. Chess Piece Occupancy

Results of the two parallel processes described previously must be fused together to determine the virtual representation of the game board. The square occupancy of the detected pieces are decided by maximizing the intersection over union (IoU) of each of the detected bounding boxes with every square found on the board. The square with the maximum IoU with the detected bounding box is determined to be the square in which the detected piece rests on and the predicted label is then marked in the 8x8 matrix representing the board. Once a square has a piece set to it, the square is no longer used for the remaining predictions as there can only be one piece per square. The resulting 8x8 matrix represeting the board state is then converted into a Forsyth-Edwards Notation (FEN) string that defines the state of the board and can be input to any chess analysis tool to virtually recreate the board.

6. Results and Experiments

The corner detection process was designed liberally such that it should always be able to find the required corner points of the squares of the board given a quality image. For every centered image in the training and validation set the algorithm finds every desired point however for some images extra points are found either between squares or on the edges of the board. For the points between squares, the angle method of determining squares filters these out as noise however the extra points found on the borders of the board can result in the extracted squares being shifted one square left and up.

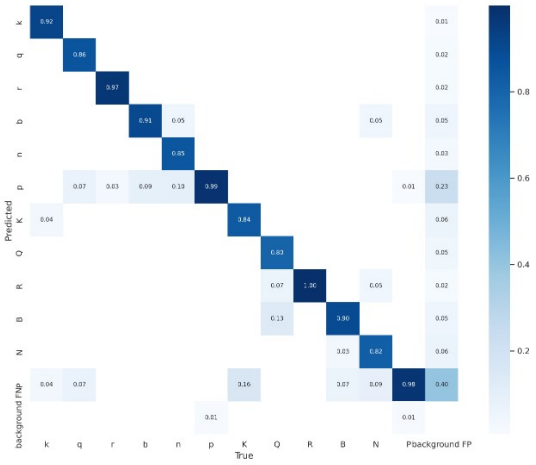

Figure 3. Confusion matrix for YOLOv3 chess piece recognition

The chess piece detector using YOLOv3 was tested against the validation set and achieved a mean average precision (mAP) of 99.4% at an IoU threshold of 0.5 for all classes of chess pieces after training for 200 epochs on the dataset described in section 4. The mAP dropped to 0.411 when varying the IoU threshold from a range of .5 to .95 at increments of 0.5. This suggests that the network is able to find the pieces in the image but is not able to give the exact defined ground truth bounding boxes. The network is able to correctly identify the label of the pieces found in the image to a high degree of accuracy as can be seen in the confusion matrix given in figure 3. The precision of the predicted labels increases with the confidence score of the prediction and reaches a high precision at a confidence value greater than 0.8 as seen in the graph in figure 4. The entire results for precision and recall of each class label can be seen in table 1. Inference of the network was tested on the NVIDIA Jetson Nano 2GB and was found to have an inference of around 1 frame per second.

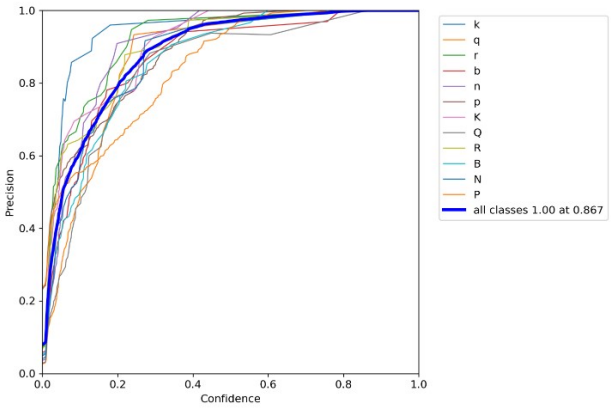

Figure 4. Precision versus confidence for YOLOv3 chess piece recognition

The square occupancy determining process was evaluated with images that provided perfect square detection with all pieces detected by the piece detector as the error from these processes propagate through the system and affect the results of the square occupancy. Given perfect squares and all detected pieces, the occupancy is determined within one square of the correct position of the physical board. Given a sequence of moves starting from the initial state of a chess game along with the legal moves available on a given turn, the proper board configuration is easily determined for most moves. Moves that allow for many possibilities such as the rook with many vertical squares or the omnidirectional queen can present a challenge. An example input image can be seen in figure 5 and the resulting digitized chess board is shown in figure 6. As it can be seen only the white queen is perfectly accurate and all other pieces are within 1 square of their correct positions.

Class	Precision	Recall	mAP@.5
all	.992	.976	.994
k	.995	.96	.985
q	.996	1	.995
r	.994	1	.995
b	.968	1	.995
n	1	.9	.995
p	1	.99	.995
K	1	.973	.995
Q	.965	.933	.991
R	.992	1	.995
B	1	.969	.995
N	.99	1	.995
P	1	.991	.995

Table 1. YOLOv3 chess piece recognition results

Figure 5. Input image of chess game state

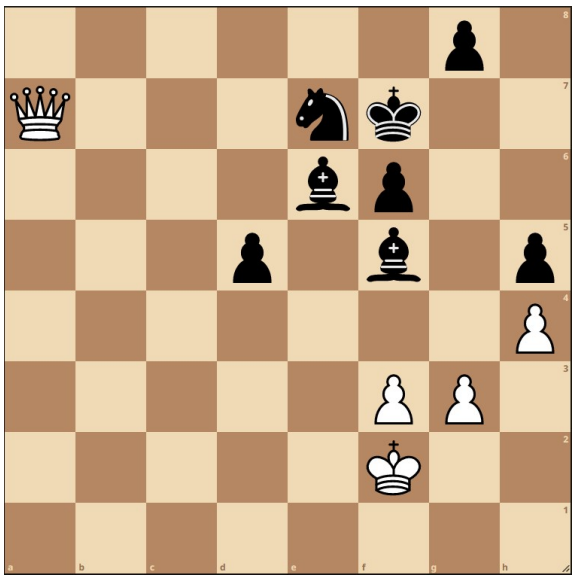

Figure 6. Digitized output board state

7. Future Works

The proposed system for logging a chess game is ultimately unfinished and the game state detection is only a single piece of the puzzle for the final product. The game state detection itself requires some fine tuning to mitigate the error for square detection and occupancy determination. As the system is intended to log an entire game from start to finish, the initial configuration of the board can be used to determine the position of the board and this information can be incorporated to prevent the edges of the board from interfering with square extraction. The same squares can be used for every turn throughout the match as the board is stationary throughout the match.

A major reason for the inaccuracy of the occupancy determination is the format of the dataset used for training the object detector. The ground truth bounding boxes given in the dataset include the entire piece and depending on the perspective of the camera the bounding box may cover more of a square that the piece is not placed on than the actual square it is on. To rectify the problem, data can be gathered from perspectives similar to what would be expected to occur in operation of the final system and the ground truth bounding boxes can be manually labelled to not cover squares that a piece is not on. This will allow for the network to give outputs closer to the actual square location such that when the IoU of the predicted bounding box and the detected squares is calculated, the maximum IoU should be the correct square that the piece is placed on.

For the complete system to operate seamlessly, each and every move that is made needs to be detected such that the new game state can be found. A simple solution to this problem would be to use a special timer that signals the system to capture a frame from the video whenever it is

pressed as each player must press the timer to indicate that their turn is over. While this solution is simple it is not ideal as it would add additional cost to the setup. Another possible solution is to use the video feed and track the gestures of the player as to detect when each player presses the timer. When the gesture for pressing a timer is detected the system can capture a frame from the video feed and begin processing the game state. Finally each game state captured needs to be compared to the previous game state captured to determine the difference in game states and give the move that caused this difference. These moves can be recorded as to give a live log of the game as it progresses and would be especially useful for broadcasting.

8. Conclusion

Digitization of a physical chess board is a complex task that requires many processes working in unison. An architecture parallelizing the square detection and piece detection processes reduces latency of game state recognition allowing for faster game modes to be analyzed. YOLOv3 performs very well for the task of chess piece recognition and achieves an astounding mean average precision of 99.4% at an intersection over union threshold of 0.5. The inference of the network on an NVIDIA Jetson Nano is quick at 1 frame per second allowing for a portable solution to logging blitz and slower game modes however the latency is slightly too high for bullet. While the piece detection performs very well on the dataset chosen, it was learned that a dataset with more precise bounding boxes in terms of the square location would provide greater results when used with the proposed parallel architecture. Given some fine-tuning, a portable system for logging the majority of chess games played over the board at tournaments is achievable

9. Github

Code for the ChessLogger system as well as the best trained YOLOv3 model for chess piece recognition can be found at the github, https://github.com/jjames71396/ChessLogger.

References

[1] Arandjelović, Ognjen. "Determining Chess Game State from an Image." Journal of Imaging 7.6 (2021): 94.

[2] Quintana, David Mallasén, Alberto Antonio del Barrio García, and Manuel Prieto Matías. "LiveChess2FEN: A framework for classifying chess pieces based on CNNs." arXiv preprint arXiv:2012.06858 (2020).

[3] Neto, Afonso de Sá Delgado, and Rafael Mendes Campello. "Chess position identification using pieces classification based on synthetic images generation and deep neural network fine-tuning." 2019 21st Symposium on Virtual and Augmented Reality (SVR). IEEE, 2019.

[4] Czyzewski, Maciej A., Artur Laskowski, and Szymon Wasik. "Chessboard and chess piece recognition with the support of neural networks." Foundations of Computing and Decision Sciences 45.4 (2020): 257-280.

[5] Wolff, Regina, et al. "Towards Computer-Vision-Based Learning from Demonstration (CVLfD): Chess Piece Recognition." J. Comput. 14.8 (2019): 519-527.

[6] Piškorec, Matija, et al. "Computer vision system for the chess game reconstruction." 2011 Proceedings of the 34th International Convention MIPRO. IEEE, 2011.

[7] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.