Live Chessboard Analysis Using Computer Vision

Zeb Zimmer
University of Minnesota, CSCI 5525, Fall 2023
https://github.com/ZebZimmer/ChessComputerVisionProject

I. PROBLEM DESCRIPTION

Since its inception chess has been played between two human competitors who use their game knowledge and cunning to outmaneuver their opponent. The steady rise of computing power in the 20th century led to a new challenger in the once human-only space. In 1996 IBM's supercomputer, Deep Blue, beat the reigning world champion, Gary Kasparov, and computers officially took over the world of chess. The technicians at IBM during the game against Kasparov needed to manually input his moves so that Deep Blue could process the board and decide what to do next. This paper aims to skip that middleman and use computer vision to process a live chess board and then collect the next best move from an external modern chess engine.

A chess set has 6 different piece types across 2 colors making 12 distinct piece types. This project detects the chess pieces present on a standard 64 square chessboard and then returns the next best move.

II. TECHNICAL ML PROBLEM

The main needs of the project are to see a chessboard and then determine if there are chess pieces present and if so classify the piece(s). These problems are best solved by machine learning models that can learn the features of a dataset which allows for detection and classification tasks that would be extremely difficult to solve with non-machine learning methods.

III. MODELS, ALGORITHMS, AND METHODS

A. YOLOv8 Model

The detection and classification problems were solved with a YOLOv8 model from Ultralytics[7] and a CNN (Convolutional Neural Network) respectively. The Ultralytics YOLOv8 is the 8th iteration of the YOLO (You Only Look Once) model[6] which was initially designed in 2015 by Joseph Redmon and Ali Farhadi at the University of Washington. Ultralytics offers YOLOv8 for free under the AGPL-3.0 License for students and enthusiasts. YOLOv8 was used due to its detection abilities on small datasets its strength in its ability to detect objects. Ultralytics designed the model with the ability to also predict a class for objects it detects but this implementation only used the detection task of YOLOv8. The classification was left for a CNN which could better learn the intrinsics of chess pieces.

B. CNN Model

The CNN was created using a Keras Sequential model from TensorFlow[4]. Three 2D convolution layers with ReLU activation and two 2D max pooling layers were used as the model architecture and an Adam optimizer with categorical cross entropy were used in training. Deep CNNs are used to learn a vast variety of features. The tasks of differentiating chess pieces did not require a deep CNN which allowed for fast training and therefore rapid prototyping. Two models of the CNN were created. One that used in-color photos with three color channels and one that used grey-scale images. This report was written with the in-color model at the forefront because the training data was given in color, and more on this later.

C. The models used together

YOLOv8 and the CNN were used in tandem to complete the chess piece detection and classification. The YOLOv8 model would take in the preprocessed image, detect the chess pieces in the image, and return the bounding box coordinates of the piece. Then for every piece detected the bounding box coordinates were used to crop the original image to just the detected piece. The detected piece was passed into the CNN for prediction after being resized to 101x46 for consistency.

Model performance was improved with minor technique changes. The live model implementation actually takes 10 images over the course of a few seconds with the intention that the user slightly perturbs the camera stand. These perturbations allow for the collective models to receive slightly varied data. The prediction of each piece and its location are saved for all 10 pictures. The final result is the strongest prediction at each square based on prediction count. The 10 images also reduces the ability of an outlier misclassification from permeating to the final board analysis.

D. Chessboard Detection and Analysis

The detection of the chessboard was done using ArUco markers[2] which works seamlessly with OpenCV[3]. The edges of the board were used in the preprocessing stage as well as in determining what square on the chessboard each detected piece lay in. An adaption of Barycentric Coordinates determined what column a piece was in and bounds checking determined the row. This process required some fine-tuning to get correct. The bottom of the bounding box prediction from the YOLOv8 model was mapped onto the board and the matching square was recorded along with the class prediction from the CNN. A mapping of the squares to the board is shown.

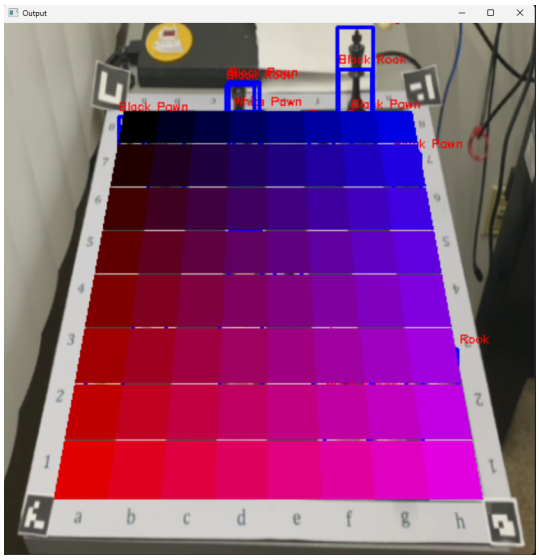

FIG. 1. Visualization of the chess squares

After the board is understood and the collection of piece predictions and locations are saved, a FEN encoding of the board is created and passed to the Stockfish API[5] to determine the next best move. A FEN encoding is a standard in the chess world that simply encodes the position of the pieces on the board.

IV. DATA AND PREPROCESSING

The data used for training is sourced from Roboflow[1] and contains 289 unique images with the labels being the bounding box coordinates for each piece in the image. The training images were taken from a small distance from the chess board and then cropped to contain only the board, this minimizes distortion at the edges of the image. After the cropping, the images were resized to 416x416 for consistency. The dataset also came with 404 extra images that were augmentations of the original 289. These extra images had varied hue, saturation, brightness, shear, and exposure to add robustness into the dataset.

When using the trained models a similar preprocessing technique was used where input images were cropped to be only the board with its pieces and then resized to 416x416. Cutting the image down increased the percentage of the image that was chessboard and had a noticeable improvement in performance. The contrast was also reduced by 15% as part of the preprocessing process but is likely a unique addition necessary for the specific lighting conditions of this paper's testing environment.

The training for the YOLOv8 model was full images of a chessboard with the locations of pieces given with bounding box coordinates and a label. When training the CNN, however, the input was just the image inside of the bounding boxes which was essentially only the piece. This allowed for the YOLOv8 to learn to detect chess pieces and for the CNN to learn to classify them.

V. RESULTS

A. Training

Training on the dataset was fast and the loss seen during training for the CNN is shown in Figures 2 and 3 with Figure 4 for the YOLOv8 model.

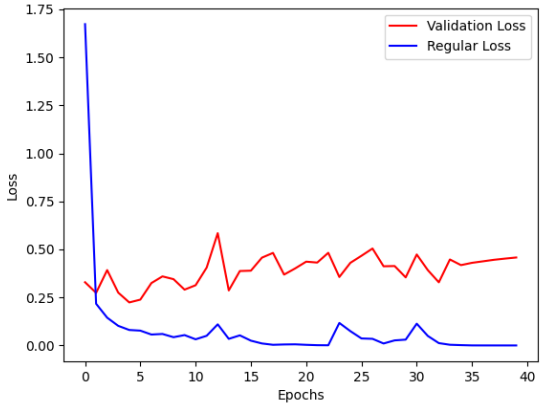

FIG. 2. In-color CNN Loss over 40 epochs

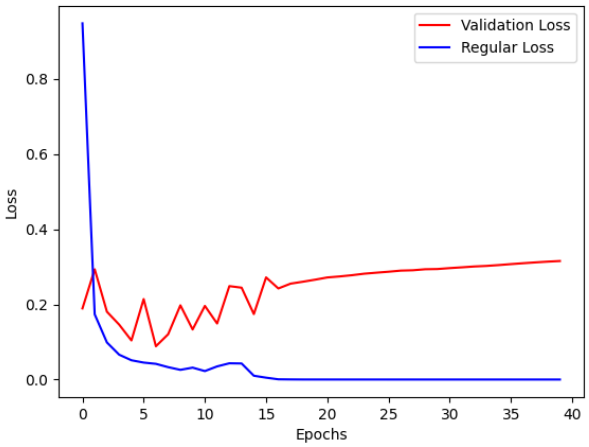

FIG. 3. Grey-scale CNN Loss over 40 epochs

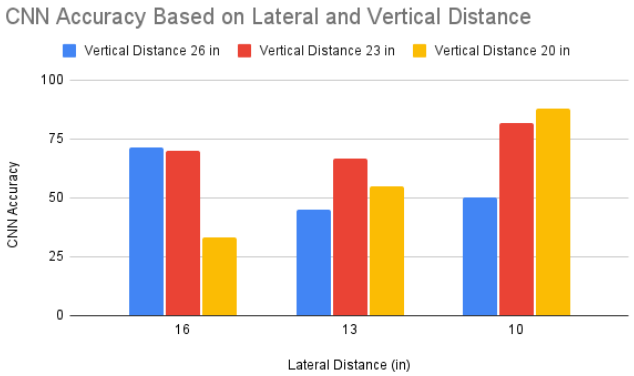

FIG. 5. In-color CNN Accuracy Based on Camera Distance to Board

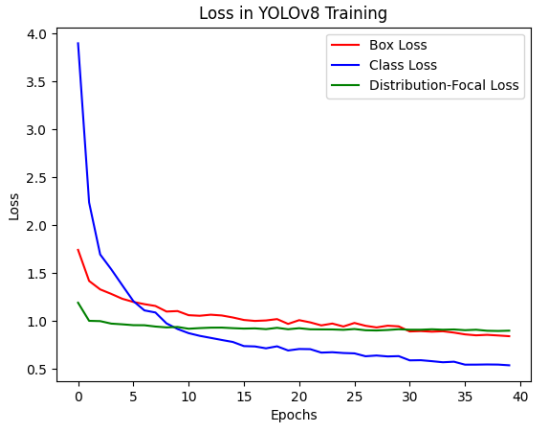

FIG. 4. YOLOv8 Loss over 40 epochs

B. Model Performance

The accuracy of the CNN is shown below based on the lateral and vertical distance of the camera to the board in inches. The results were one-shot and involved no post-processing nor averaging techniques outlined in section III.

The YOLOv8 had an average of 91% accuracy over all 9 distance combinations. At its best it was 100% and at its worst 86%.

The results came from the board configuration shown below. This setup was designed to test the limits of the models.

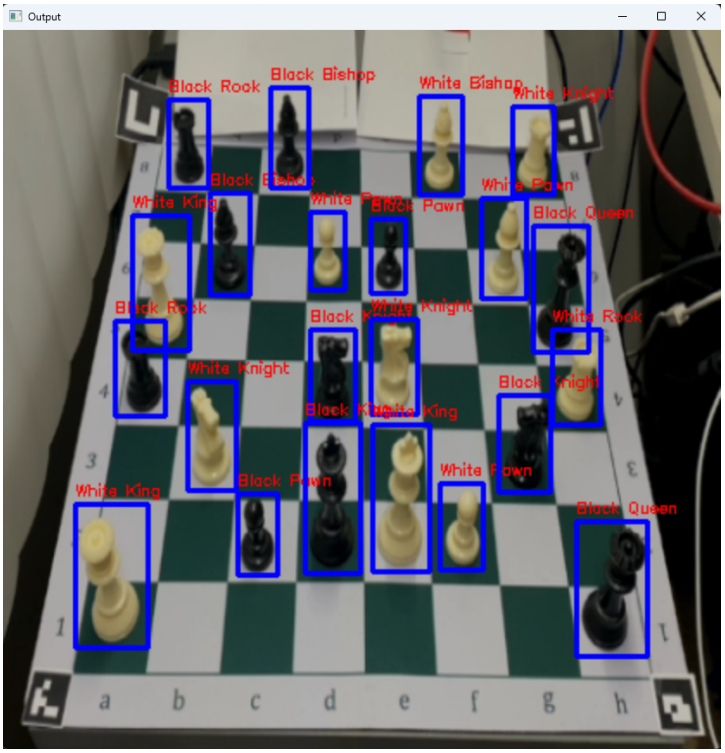

FIG. 6. Bounding Box Prediction with Piece Prediction on the Test Configuration

C. Real Position with Best Move

Figure 7 shows a real chess position that any player could find themselves in. The white pieces are under

attack and it could be easy for a player to make a big mistake. Using the models the player would learn that the best move is C2A2 which moves the white rook and checks the king and it leads to checkmate.

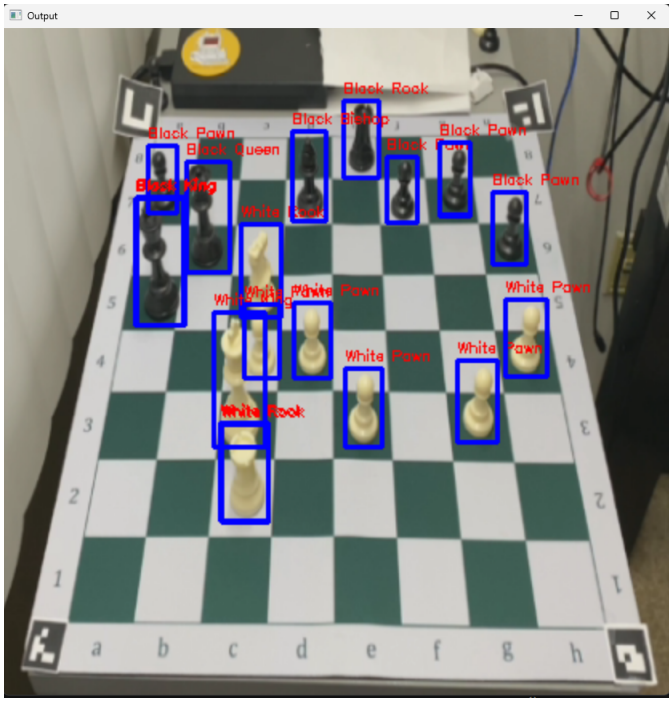

FIG. 7. Board State with the given best move of "c2a2"

VI. DISCUSSION

A. Training

A large challenge with this project was overcoming the dataset. Having 292 unique images with 2894 total labels was sufficient to train the YOLOv8 model but the CNN needed more data. The supplementary 404 augmented images helped significantly but still the training of the CNN converged extremely quickly. Testing at various epochs (namely 10, 20, 30, and 40) showed that 40 epochs was the best. At 10 epochs the CNN determined that most pieces were a Knight. At 30 epochs bishops, rooks, and pawns were mixed up constantly. At 40 epochs the CNN was at its best. The speculation is that the CNN was overfitting to the training data but because the conditions of the live testing were made to be as close to the conditions of the training data as possible the results were favorable.

The YOLOv8 model continued to see productive learning deep into its training. Testing with various epochs (again 10, 20, 30, and 40) showed that with 40 epochs of training the model had the best detection abilities. The YOLOv8 model has a mode that will also classify. The classification results were poor, separate from the success of the detection. The lacking classification ability of the YOLOv8 model with the given dataset was the main reason for the dual model setup. The CNN could focus on classification while the YOLOv8 model could focus on detection.

B. Model Performance

Figure 5 shows the variability of the CNN accuracy at various positions relative to the chessboard. Measured from the closed side of the board to the camera, testing was done at 3 different distances in the horizontal direction and then at 3 different heights. The desired takeaway from the graph is that the CNN did not generalize well to different camera positions. This is due to the singular camera position used in the training. The chess piece configuration used in testing (Figure 6) was designed to test the limits of the models' ability. For example, the queens in the corner of the image received the worst of the elongation effect of being nearest to the camera and suffered the most misclassifications in testing. The bishops and rooks in the back row were the second most common source of error as similar-looking pieces deeper in the image suffered from misclassifications as well.

C. Improving Performance

To improve the performance of the CNN the lighting conditions and framing of the chessboard were made to be as close to the training data as possible. This involved testing various camera positions, cropping schemes, and two different cameras. The change that had the most significant improvement in performance was switching to an iPhone 11 camera from a 1080p Logitech webcam. This camera improvement increased the resolution of the photos and allowed for the second most significant improvement. That change was moving the camera further away from the board, possible due to the increase in resolution, and cropping the image down to be only the board. This reduced the distortion effects of pieces near the edge of the picture. The iPhone 11 also allowed for customizations of the output image like the 15% reduction in contrast which improved classification.

The CNN model was also altered to be trained on and then made predictions on black and white images. This reduced the number of classes to 6. This model had equivalent and possibly slightly better accuracy than the CNN trained with color images. The grey-scale CNN still struggled in the same aspects as its in-color version where elongated pieces at the near corners were generally misclassified. The loss for the grey-scale CNN had less validation loss and has more potential due to the smaller number of classes. The in-color model was

chosen to base the report because the training images were in-color by default and the augmentations took advantage of that fact.

The strategy of taking 10 pictures, as mentioned in III.c. improved performance by allowing for an averaging of predictions which reduced outlier classifications. This benefit was furthered by creating small perturbations in the images (mainly by moving the camera slightly). These minor changes helped cement the correct classifications. Figure 7 shows a misclassification of the knight on C7 to be a white rook. The average of the 10 pictures, however, came out to be a white knight. This allowed for the correct best move to be found from the external chess engine.

D. Best Move Prediction

The most successful part of the project, aside from the YOLOv8 detection, was the mapping of predicted pieces to the chessboard so that an external chess engine could determine the next best move. A nontrivial amount of time was spent determining the best way to determine the location of the chess squares in the image. Ultimately, ArUco markers were used because they simultaneously made it possible to determine square positions but also made cropping the image easier. The board state was sent to Stockfish which sent back the next best move based on the current engine's capabilities.

VII. NEXT STEPS

There are a few paths to take this project. The most likely would be adding a simple way for the user to edit the predictions. This slightly defeats the purpose of using computer vision but it would still allow the project to be a time saver if trying to run analysis on a live board quickly. Only a few pieces would need to be added or changed from the models' prediction instead of having to do the entire board.

Another more fruitful path would be to greatly increase the size of the dataset with images that also vary the distance and height of the camera. The main benefit from these additional images would be to squeeze the last bit of performance necessary to get the CNN classification to near 100% but would also allow for solid predictions from the models with less attention needing to be paid to the conditions (lighting and angle).

The most likely next step will be a reduction of the testing code into a GUI that simplifies the user experience. Right now the code is a bit convoluted for a new user to try and use.

VIII. FEEDBACK

The feedback that was most helpful was conversations with the Professor during office hours. They were small brainstorming sessions where the Professor offered ideas on how to tackle the problem and improve results. There was a comment from a peer on the initial project proposal who suggested using a trained model as either an upstream base or as the main CNN. The search for a decent model to aid in the detection of the pieces led to the inclusion of the YOLOv8 model into the project as the detection method.

There was peer feedback that suggested classifying only 6 pieces grey-scale and using basic color checks to determine the color of the piece later. The feedback arrived a bit late in development but was rather clever. It took a bit of altering of the model and needed a retraining run but the results were a slight improvement in classification. As mentioned in VI.c. the paper was written based on the in-color version because the training data's augmentation took advantage of the in-color characteristic of the training data. The YOLOv8 model was not trained on grey-scale because the model was a black box and performance was more than sufficient on simply the in-color data.

IX. AUTHOR'S CONTRIBUTION

This will be a brief list of what the author did in terms of coding for the project. All of the architecture to take an image, preprocess it, feed it to the models, postprocess the image (averaging), map the pieces from the image to the chess board, and send the board state to an external chess engine was created by the author. Also, the code necessary to train CNN (mainly preprocessing images) came from the author. The CNN model, while basic, is the author's code and was changed slightly through the design process. The YOLOv8 model needed to be trained by the author but the functions and architecture was of Ultralytics design.

X. REFERENCES

[1] RoboflowChess Pieces Object Detection Dataset. Chess pieces object detection dataset. 2021. Roboflow. https://public.roboflow.com/object-detection/chess-full/24 Accessed: 11/28/2023

[2] arucoGarrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. Marín-Jiménez, M. 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition4762280-2292. https://www.sciencedirect.com/science/article/pii/S0031320314000235 doi: https://doi.org/10.1016/j.patcog.2014.01.005

[3] OpenCV2023Dec. https://opencv.org/

[4] TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/Sequential

[5] StockfishOfficial-Stockfish. . Official-Stockfish/stockfish: A free and strong UCI Chess Engine. Official-stockfish/stockfish: A free and strong uci chess engine. https://github.com/official-stockfish/Stockfish

[6] YOLORedmon, J., Divvala, S., Girshick, R. Farhadi, A. 2016. You Only Look Once: Unified, Real-Time Object Detection. You only look once: Unified, real-time object detection.

[7] UltralyticsUltralytics. . Ultralytics YOLOv8 Github. Ultralytics yolov8 github. https://github.com/ultralytics/ultralytics