Chess Proctor Application

Abdelazim Lokma, Risheet Nair, Zachary Meeks

Problem Statement: The objective of this project was to develop a computer vision system capable of automatically detecting and recognizing chess moves from a video recording of a chess match. The system aims to accurately identify each move made by the players throughout the game and report the sequence of moves at the conclusion of the match.

Introduction: The analysis and documentation of chess games play a crucial role in understanding player strategies, learning from past games, and advancing the state of the art in chess theory. However, the manual process of recording and transcribing moves from a chess match can be laborious and time-consuming.

In response to these challenges, this project attempts an approach to automate the process of detecting and recognizing chess moves from video recordings of chess matches. The objective is to develop a computer vision system capable of accurately identifying each move made by the players throughout the game and reporting the sequence of moves at the conclusion of the match.

The significance of this project lies in its potential to revolutionize the way chess games are analyzed and documented. By harnessing the power of computer vision techniques, the system aims to streamline the process of move detection, providing a more efficient and reliable method for capturing and preserving the intricacies of chess gameplay. Subsequent sections will delve into greater detail regarding the specific techniques and algorithms employed and the evaluation of system performance.

Data Aggregation and Annotation: For the purpose of training our models to detect the corners of the chessboard and to identify chess pieces, our initial approach involved exploring pre-existing datasets. However, upon thorough examination of available datasets, we encountered significant inconsistencies in labeling mechanisms, rendering them unsuitable for our purposes. Even datasets sourced from platforms like Roboflow lacked the consistency required for effective utilization.

Subsequently, we decided to capture our own video footage of a chess game in progress. This approach ensured a tailored dataset that aligned closely with our specific requirements. The video footage encompassed various angles and lighting conditions to capture a diverse range of scenarios akin to real-world chess matches.

The captured video frames served as the foundation for our training images. Leveraging Roboflow, we performed manual annotation to label the corners of the chessboard and the individual chess pieces

within each frame. This manual annotation ensured precise and accurate labeling, essential for training robust models capable of reliably detecting chessboard corners and identifying the pieces.

To enhance the diversity and robustness of our dataset, we augmented the annotated data by introducing additional variations. This included augmenting lighting conditions to simulate different environments and applying small rotations (±2 degrees) to the frames to account for potential positional variations of the chessboard and pieces.

The resulting dataset served as the cornerstone for training our computer vision models. By curating our own dataset tailored to our specific needs and meticulously annotating it, we ensured the quality and consistency required for the successful development of our automated chess move detection system.

Methods: In our pursuit of developing a robust system for automated chess move detection from video, we leveraged the YOLO (You Only Look Once) model architecture. Specifically, we employed YOLOv8, a variant of the YOLO model, renowned for its efficiency and effectiveness in object detection tasks.

To train the YOLOv8 model for detecting chessboard corners and identifying chess pieces, we adopted a straightforward approach. Utilizing the meticulously annotated dataset of chessboard corners and piece images, generated through manual annotation as described in the preceding section, we fed this data into the YOLOv8 training pipeline.

The YOLOv8 model was trained using the annotated images, with the task of learning to accurately localize and classify chessboard corners and individual chess pieces within each frame. Notably, the YOLOv8 architecture's inherent design facilitated seamless integration of the annotated dataset into the training pipeline, requiring minimal preprocessing or customization.

The YOLO algorithm revolutionized object detection by moving away from region based detection methods such as R-CNN, to single pass end to end neural nets. YOLO is able to directly classify an object in a single pass. YOLOv8 works by dividing the input image into a grid of cells, and predicting a set of bounding boxes and class probabilities for each cell. It then uses Non Maximum Suppression to eliminate duplicate detections.

The training process involved iterative optimization of the model's parameters to minimize the detection error and maximize accuracy in identifying chessboard corners and pieces. YOLOv8's efficient training scheme enabled rapid convergence towards optimal performance, significantly expediting the model development process.

By harnessing the capabilities of the YOLOv8 model and leveraging our meticulously annotated dataset, we successfully trained a robust object detection system capable of accurately detecting chessboard corners and identifying chess pieces within video frames. This trained model served as a pivotal component of our automated chess move detection system,

contributing to its overall efficacy and reliability in real-world applications.

Upon successfully training the YOLOv8 model for detecting chessboard corners and identifying chess pieces, our focus shifted towards implementing the logic for detecting chess moves from video footage of a chess match.

The process commenced by utilizing the trained board corner detection model at the outset of each video. This initial step served to establish the precise location and orientation of the chessboard within the video frame, providing a crucial reference point for subsequent move detection. Subsequently, we devised a systematic approach to traverse through the chessboard tiles at regular intervals within the video frames. Utilizing the established board corner coordinates, we iterated over each tile of the board, analyzing its contents every few frames.

For each tile examined, we applied the trained piece detection model to determine the presence of a valid chess piece and, if detected, identify the specific type of piece occupying the tile. This iterative process enabled real-time monitoring and tracking of the chessboard configuration throughout the duration of the video.

To ensure the accuracy and reliability of piece detection, we implemented a motion detection mechanism to skip frames where hand movements were detected. Hand movements often introduced artifacts and disturbances that could compromise the accuracy of piece predictions.

Additionally, a perspective-transformed image was utilized to enhance piece detection accuracy on each tile. By transforming the board from the observed angle to an actual square perspective, we minimized distortions and improved the model's ability to accurately identify pieces.

During the training phase of the piece detection model, a masking technique was employed to focus the model's learning on the specific tile of interest. By masking everything except the tile being trained on, the model was better able to learn the nuances and characteristics of the pieces within that particular context.

Utilizing the information gleaned from the board corner detection and piece identification models, we systematically compiled a list of detected moves, adding newly identified moves to the list as they occurred within the video footage.

This methodological approach enabled the automated detection and recording of chess moves from video recordings of chess matches, paving the way for the development of a robust and efficient system for analyzing and documenting chess gameplay in real-time scenarios.

Design of Program: The design of the program went through a couple iterations, but essentially remained faithful to the original design throughout.

Our solution fundamentally boils down to three responsibilities:

1) Recognize the board
2) Recognize the pieces
3) Backend accounting logic

We allocated these three responsibilities to three classes:

1) ChairmanOfTheBoard
2) PieceWizard
3) Accountant

Our first attempt at a solution consisted of ChairmanOfTheBoard recognizing the board and doing the calculations of where all 64 chess squares exist on the original board space as well as which squares probably had a piece on them. In this version, ChairmanOfTheBoard had two exposed methods. The first would return a list of lists of all of the grid coordinates in the original space. The second would return a list of all of the indexes that looked like they might have a piece on them (so as to potentially avoid the more expensive calculations that PieceWizard would do).

Accountant would then compare the list of squares with possible pieces on them to the definite list of pieces in the current game-state. If the squares matched, Accountant wouldn't do anything. When Accountant noticed a discrepancy, it would ask PieceWizard if the piece was a real piece or not at the squares in question. PieceWizard in this model would have been trained to recognize unknown values such as a piece in motion or a hand in the frame (basically it would need to be very very good at distinguishing between pieces, empty-pieces, and not pieces).

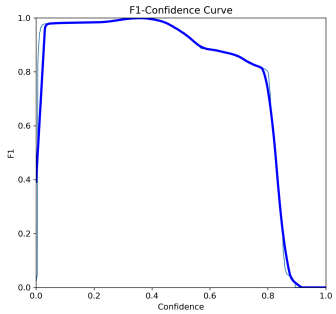
During implementation, we didn't like having to deal with the coordinates of a non-square quadrilateral and so we applied a matrix transformation to transform the board into perfect square-space. We also decided that from a training-time perspective, it made sense to skip the responsibility of having ChairmanOfTheBoard return which squares might have a piece on them, since that would require extra training while not providing functionality that we couldn't otherwise get from PieceWizard. The updated design sees ChairmanOfTheBoard responsible for transforming the board into perfect squarespace and finding the coordinates of the transformed board with padding and returning that same image to Accountant. Accountant then iterates over all of the chess squares, and first checks if there has been motion or not (as we had an issue getting the PieceWizard model to identify unknowns properly) and only if there has been no recent motion in the vicinity of the transformed image, then we iterate over the chess squares until we find an empty square that wasn't empty in the previous board-state and then search the board for that piece and record the move.

PieceWizard's responsibilities from the beginning was always just to check if a grid contains a piece or not, and if so which piece. Originally PieceWizard checked for unknowns but our motion-detector now handles that.

The motion detector transforms the rgb image into grey-scale and looks for all pixel values whose absolute differences are greater than 25 and counts all of them.

Results:

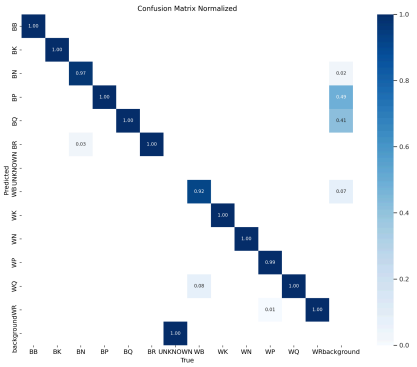
We visualized the corner detection performance via an F-1 curve as seen here. We concluded that the model worked reasonably well. However, the model was susceptible to changes in the orientation of the board, as well as cases where the corners were completely occluded. The



F1-Confidence curve shows that the model operates best at a confidence threshold of 40%, while this indicates that the model is not certain of its predictions, the F1 score

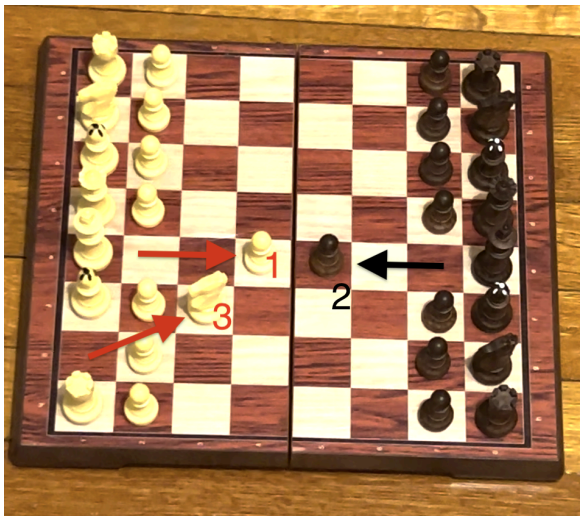
that it achieves at this threshold is quite high. Given that the chessboard is assumed to be static for the duration of the recording, only one correct detection is needed to locate the board and calculate the grid coordinates‘ positions.

YOLO Piece Detection Results:



On the other hand, the YOLO piece model’s performance was essential to the performance of our overall program. Originally, the model greatly struggled to correctly classify certain pieces, due to the relatively high angles we needed to film chess matches from (in order to see all four corners of the chessboard), this reduced the feature differences between certain pieces (such as bishops and pawns, which look similar when viewed from a top down perspective). The model also greatly suffered from small changes in lighting, so in order to fix this, we created an augmented dataset that created variations of the training data with different brightness levels. These augmentations greatly improved model performance, giving us a near perfect confusion matrix of classifications. However, even a near perfect model performance could (and did) cause issues in the final program.

Chess Proctor Performance:




```
Game over. Chess moves were:
MOVE MADE: 1. WPe4
MOVE MADE: 1. BPe5
MOVE MADE: 2. WNf3
```

Final program performance was adequate, we had some games that showed successful tracking, other games were less successful. The image displayed above is a snapshot from a short video that captures a sequence of three chess moves. Our program successfully recorded and displayed all three moves depicted in the correct order of their occurrence.

However, other recorded games were less successful, where some moves were either not detected at all, or detected in an incorrect order. We believe that the cause of this poor performance could be generated by two possible sources; firstly, our motion detection system may have been too sensitive, ignoring potentially crucial frames of the video where a move could have taken place. Secondly, during piece capture moves, we faced challenges in accurately determining the subsequent position of the

capturing piece, as our YOLO piece classifier occasionally misclassified pieces.

Conclusion: Our study delved into the development of an automated chess move detection system utilizing computer vision techniques. Through rigorous experimentation and analysis, we obtained valuable insights into the performance and limitations of our implemented models.

The evaluation of our corner detection model revealed promising results, as depicted by the F-1 curve. While the model exhibited reasonable performance, it demonstrated susceptibility to changes in board orientation and occlusion of corners. Despite these challenges, the model's ability to accurately detect corners, even at a confidence threshold of 40%, underscored its utility in establishing the chessboard's position and calculating grid coordinates.

The pivotal role played by the YOLO piece detection model in our system's performance cannot be overstated. Initially encountering difficulties in correctly classifying certain pieces due to filming angles and lighting variations, we successfully addressed these challenges through the creation of an augmented dataset. These augmentations significantly improved model performance, resulting in near-perfect classification accuracy and a highly reliable confusion matrix.

While the final program exhibited adequate performance in successfully tracking and recording chess moves in some games, it encountered challenges in others. Notably, issues arose during piece capture moves, where the classifier struggled to accurately identify the subsequent position of the capturing piece. Additionally, the sensitivity of our motion detection system potentially led to the omission of crucial frames containing moves.

Despite these challenges, the system demonstrated its capability in accurately recording and displaying sequences of chess moves, as evidenced by successful outcomes in several games. However, the occasional discrepancies in move detection highlight areas for further improvement, particularly in refining motion detection algorithms and enhancing piece tracking accuracy.

Overall, our study contributes valuable insights and methodologies towards the development of automated systems for analyzing and documenting chess gameplay, laying the groundwork for future advancements in this domain. Through continued refinement and optimization, we anticipate further enhancements in system performance and reliability, ultimately facilitating a more seamless and accurate analysis of chess matches.