Retrieving Forsyth Edwards Notation from Live Chessboard Using YOLOv11

First Dr. K Deepak[1*], Second P.V. Adithiyan[1†], Third G V Krishna Kumar[1†], Fourth Jeevan Sendur G[1†], Fifth Rahul K[1†]

[1*]Dept. of Computer Science and Engg., Amrita School of Computing, Amrita Vishwa Vidyapeetham, Street, Chennai, 601103, Tamil Nadu, India.

*Corresponding author(s). E-mail(s): k_deepak@ch.amrita.edu;
Contributing authors: adithiyan999@gmail.com;
kalyangvkk2004@gmail.com; jeevansendur8905@gmail.com;
rahulbio789@gmail.com;
†These authors contributed equally to this work.

Abstract

Chessboard recognition and position analysis are critical components of automated chess game recording and analysis. This paper presents a novel deep learning-based pipeline for real-time extraction of Forsyth-Edwards Notation (FEN) from chessboard images. Our approach involves a YOLOv8 segmentation model, fine-tuned on a custom-labeled dataset of 300 images using Roboflow, to detect the chessboard and extract its boundaries. Once detected, the board is warped to a fixed perspective to ensure consistent alignment. The board is then divided into an 8×8 grid, where each cell represents an individual chess square. We employ a YOLOv11 classification model to recognize chess pieces within each grid cell, achieving a classification accuracy of 95.4%. The detected board layout is then converted into FEN notation, enabling seamless integration with digital chess applications. Our method demonstrates high reliability in real-world conditions, facilitating applications in game broadcasting, chess tutoring, and historical game digitization. Experimental results validate the robustness of our approach, making it a significant step toward real-time, automated chess position recognition.

Keywords: Computer Vision, Deep Learning, YOLOv8 Segmentation, YOLOv11 Classification, FEN Notation Extraction, Chessboard Recognition

1 Introduction

Chess has long been regarded as one of the most intellectually demanding games, requiring strategic thinking, pattern recognition, and deep analysis. With the widespread adoption of digital technologies, the integration of artificial intelligence and computer vision into chess has gained significant attention. One of the most challenging aspects of digitizing chess is automating the recognition of board positions from real-world images. This task is crucial for various applications, including automated game tracking, real-time tournament broadcasting, historical game digitization, and AI-driven chess coaching. Manually recording chess moves is time-consuming and prone to errors, making automation a necessity in modern chess environments. While chess engines like Stockfish and AlphaZero have revolutionized gameplay analysis, they still rely on manually inputted board states. Automating the extraction of chess positions from images and converting them into Forsyth-Edwards Notation (FEN) would enable seamless interaction between physical and digital chess systems, enhancing accessibility, efficiency, and accuracy.

Existing methods for chessboard recognition rely heavily on classical computer vision techniques such as edge detection, Hough Transform, and contour-based board identification. While effective in controlled environments, these methods struggle when faced with real-world challenges such as varying lighting conditions, different board orientations, occlusions, and diverse chess set designs. Variability in board materials, camera angles, and surrounding objects further complicates the extraction of board positions. Additionally, recognizing chess pieces presents another layer of complexity due to their often similar silhouettes and small distinguishing features. Traditional approaches that depend on hand-crafted image processing techniques lack the robustness required for accurate recognition across diverse datasets. These challenges necessitate a deep learning-based approach that can generalize effectively across different chessboards, piece styles, and environmental conditions.

This research addresses the problem of automated chessboard recognition by proposing a deep learning-driven pipeline capable of detecting chessboards, identifying individual squares, recognizing piece positions, and converting the extracted information into FEN notation. The ability to automate this process has far-reaching implications, particularly in live tournament broadcasting, where accurate and instantaneous board state updates are essential for commentators and spectators. In chess education, automated position recognition can facilitate real-time feedback, allowing students to analyze their games without manual input. Additionally, historical chess matches, often recorded in books or older formats, can be digitized with minimal effort, preserving valuable game data for future analysis. AI-based chess training systems can integrate this technology to improve gameplay evaluation and strategy formulation, providing players with data-driven insights. The impact extends beyond competitive play, as this system can also be used to assist visually impaired individuals by integrating real-time speech-to-text or haptic feedback mechanisms, enhancing accessibility in chess.

To achieve robust and scalable chessboard recognition, our approach leverages deep learning techniques for board detection, perspective correction, and piece classification. Unlike conventional methods, which require predefined rules and extensive

feature engineering, deep learning models can learn directly from data, making them adaptable to real-world variations. The first step in our methodology involves detecting the chessboard within an image using a fine-tuned YOLOv8 segmentation model. By training on a diverse dataset of 300 annotated chessboard images, this model can accurately identify board boundaries under different environmental conditions. Once the board is detected, a perspective transformation is applied to warp it into a fixed reference frame, ensuring uniformity in further processing. This step is crucial as varying camera angles can distort the board structure, leading to inaccurate piece classification. After correcting the perspective, the board is divided into an 8×8 grid, with each square extracted separately for classification.

Recognizing chess pieces within these squares presents a significant challenge due to variations in size, lighting, and design. To address this, we employ a YOLOv11 classification model trained specifically to distinguish between different chess pieces and their respective positions on the board. Unlike traditional classification models that require extensive preprocessing, this deep learning-based approach enables robust piece identification with minimal manual intervention. The final step involves converting the extracted board state into FEN notation, a standard format used for chess game representation. This conversion allows seamless integration with chess engines, enabling AI-based analysis, automated game tracking, and real-time move suggestions. By combining segmentation, transformation, and classification models, our system provides an end-to-end solution for chessboard position recognition, eliminating the need for manual tracking and ensuring high accuracy across different board configurations.

The significance of this research extends beyond chess. The methodology developed in this study, particularly the combination of deep learning-based segmentation and classification, has broader applications in structured grid-based recognition problems. Other board games such as Go, Checkers, and Shogi require similar position-tracking mechanisms, making our approach adaptable for multiple game analysis applications. Additionally, the underlying techniques for board detection and piece classification can be extended to other domains, including table-based document recognition, automated surveillance, and real-time object detection in industrial settings. The ability to identify structured patterns from images efficiently has implications in robotic vision, autonomous navigation, and intelligent retail systems where object localization and classification play a crucial role.

The remainder of this paper is structured as follows. The Methodology section provides an in-depth explanation of our approach, covering dataset preparation, model architectures, training processes, and the complete inference pipeline. This section elaborates on the deep learning techniques employed for board segmentation, perspective correction, and piece classification, detailing how each step contributes to the overall recognition pipeline. The Results and Discussion section presents an evaluation of our model's performance across different datasets, analyzing its robustness under various conditions and discussing the challenges encountered. Additionally, potential improvements and future research directions are explored to further enhance the scalability and efficiency of the system. Finally, the Conclusion summarizes the key findings of our research and highlights its contributions to the field of automated chess analysis and AI-driven game tracking.

By addressing the challenges associated with chessboard recognition through deep learning, this research contributes to the advancement of AI-powered chess applications, game analytics, and broader computer vision tasks. The proposed solution enhances the efficiency of chess position extraction, offering practical benefits in real-time tournament tracking, historical game digitization, and AI-based training systems. As chess continues to evolve in the digital era, automation-driven approaches like the one proposed in this study will play an increasingly vital role in bridging the gap between physical and digital chess, ensuring accessibility, accuracy, and seamless interaction with modern AI technologies.

2 Related Works

Azlan Iqbal (2022) proposed a method to update Forsyth-Edwards Notation (FEN) in chessboard character strings without the need for intermediate array representations. This research advances the field of accurate and effective board state management for chess and comparable games in scenarios when array-based components are unavailable or impractical. It's interesting to note that the technique ensures immediate export readiness and supports extra processing requirements. To demonstrate its efficacy, examples of its skill in a range of game scenarios are given, such as pawn promotion, en passant, and castling.

Debasis Ganguly et al. (2024) examined the retrieval of chess game positions that are similar to a given query position from a collection of recorded games from the perspective of information retrieval (IR). Their IR-based approach makes use of the reversed arrangement of cached chess positions for efficient retrieval. Rather than providing exact matches, this method offers approximate search functionality. The chess pieces' placement, reachability, and connectivity are used to determine similarities, and each game state is encoded with a textual representation. The effectiveness of their similarity computation method was demonstrated when they generated a brand-new evaluation benchmark dataset and produced MAP and nDCG values of 0.4233 and 0.6922, respectively.

David Mallasen Quintana et al. (2020) tackled the enormous technological task of automatically digitizing chess games using computer vision. Their investigation centers on the interest chess engines have generated in over-the-board (OTB) game analysis and broadcasting among players and tournament organizers. While previous methods have demonstrated potential, realistic and economical implementation still depends on increasing recognition accuracy and reducing latency. The team successfully tested these techniques on a single-board Nvidia Jetson Nano computer. In addition to expediting the chessboard detection method and examining many Convolutional Neural Networks for chess piece classification, they successfully mapped the chess pieces on the embedded platform. Among their notable achievements are a working framework with a 92% piece classification accuracy, a 95% board detection accuracy, and the ability to digitize a chess position in less than a second from an image.

The important problem of chess piece recognition and positioning for chess robots was addressed by Yongfeng Yang et al. (2022). They presented a method that combines

maximum connected component centroid localization with convolutional neural network (CNN) identification. The segments were initially pre-segmented using the HSV color space in order to obtain the greatest number of related components. After then, the segmentation results were dilated and deteriorated. The position of these components was ascertained using their centroid coordinates. After that, a trained CNN was used to identify the chess pieces. This CNN used techniques to lower the amount of parameters while ensuring high recognition rates for deployment on low-resource devices. With a 98.7% chess piece recognition accuracy, the average placement error and time on a 28 cm by 28 cm chessboard are 0.48 mm and 11 ms, respectively.

In their 2019 work, Delgado Neto et al. investigated chess piece recognition using computer vision. Numerous strategies were used to tackle the issue, with differing degrees of effectiveness and complexity. Deep learning, a cutting-edge technique for image recognition, typically requires big datasets. The study examines how to use Blender's Python API to detect synthetic chess images and optimize the VGG16 convolutional network to achieve piece recognition accuracy of over 97%. Potential uses include the ability to record actual chess games automatically and to allow internet participants to play in real time with actual boards.

3 Methodology

This section outlines the methodology for automating chessboard recognition and FEN notation extraction using deep learning. The proposed pipeline consists of multiple stages, including dataset preparation, data preprocessing, chessboard detection using YOLOv8 segmentation, perspective transformation, grid extraction, piece classification using YOLOv11, and final conversion to FEN notation. The following subsections detail each stage.

3.1 Data Visualization

To analyze dataset diversity, we plotted sample images showing different lighting, angles, and backgrounds. Figure **??** depicts some samples of the dataset.

The dataset's class distribution is visualized in Figure 2, ensuring balanced representation across different scenarios.

3.2 Dataset and Preprocessing

The effectiveness of deep learning models depends significantly on the quality and diversity of the training dataset. For chessboard detection, we collected and annotated a dataset consisting of 300 images that capture various chessboard configurations under different lighting conditions, angles, and backgrounds. To ensure generalization, the dataset incorporates boards from multiple sources, allowing the model to handle diverse real-world scenarios.

To enhance robustness and mitigate overfitting, various data augmentation techniques were applied. Random rotations ranging from -20° to 20° were introduced to account for slight tilts in real-world captures. Scaling transformations within a factor of 0.8 to 1.2 helped the model generalize to different board sizes and camera

Fig. 1: Few Samples of the Chessboard Images of our dataset

distances. Brightness and contrast adjustments were performed to simulate varying lighting conditions, ensuring that the model remains invariant to changes in illumination. Additionally, Gaussian noise was added to images to improve robustness against real-world distortions such as compression artifacts and sensor noise. Furthermore, perspective transformations were applied to introduce slight warping in the images, making the model more resilient to viewpoint variations.

This augmented dataset was then used to fine-tune the YOLOv8 segmentation model for chessboard detection and the YOLOv11 classification model for identifying individual chess pieces, ensuring reliable and accurate performance across diverse input conditions.

3.3 Chessboard Detection using YOLOv8

Chessboard detection is the foundational step in the recognition pipeline, as all subsequent processes rely on accurately localizing the board within an image. We employed YOLOv8, a state-of-the-art object detection and segmentation model, to detect and segment the chessboard. Unlike traditional computer vision techniques that rely on edge detection and contour analysis, YOLOv8 leverages deep learning to achieve higher robustness across varying environmental conditions such as lighting changes, occlusions, and perspective distortions.

The model was trained on our annotated dataset, where each chessboard was labeled using polygon masks to define its precise boundaries. The dataset was curated to include images captured under diverse conditions, including different camera angles, resolutions, and board designs. To ensure effective learning, the model was fine-tuned

Fig. 2: Class distribution across the dataset.

using transfer learning, leveraging pre-trained weights from COCO and adapting them to the specific task of chessboard detection.

YOLOv8 generates an output matrix of size $N \times 6$ for each detected object:

$$D = \{(x_i, y_i, w_i, h_i, c_i, s_i)\}, \quad i = 1, 2, \ldots, N \tag{1}$$

where (x_i, y_i) represents the center coordinates of the bounding box, (w_i, h_i) denote the width and height of the detected region, c_i is the predicted class label (chessboard), and s_i is the confidence score assigned to the detection.

To ensure the highest accuracy in chessboard detection, we implemented a selection strategy where the detection with the highest confidence score was chosen as the chessboard region. In cases where multiple detections were present, non-maximum suppression (NMS) was applied to eliminate redundant overlapping detections and retain only the most confident bounding box. The selected bounding box was further refined by performing a contour analysis within the detected region, ensuring that the extracted board edges align accurately with the real chessboard boundaries.

Once the board was detected, we extracted the region of interest (ROI) from the input image and passed it forward to the perspective transformation module. This ensures that the board is consistently processed in the subsequent steps, irrespective of its initial orientation or placement within the image.

3.4 Perspective Transformation

Since chessboard images can be captured from arbitrary angles, the detected board often appears distorted due to perspective projection. If left uncorrected, this distortion can introduce inconsistencies in subsequent processes, such as grid segmentation and piece classification. To address this, we apply a perspective transformation that warps the detected board into a fixed top-down view, ensuring uniformity across all images.

The perspective transformation is computed using four corner points of the chessboard, extracted from the YOLOv8 segmentation mask. Let these detected corner coordinates in the input image be denoted as:

$$P_{\text{source}} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\} \tag{2}$$

where each (x_i, y_i) corresponds to the estimated positions of the four extreme corners of the board.

To rectify the board, we define a standard reference frame where the chessboard is aligned perfectly in an 8×8 grid. The target coordinates are set as:

$$P_{\text{target}} = \{(0, 0), (S, 0), (S, S), (0, S)\} \tag{3}$$

where S represents the standardized board size in pixels.

A homography matrix $\mathbf{H}$ is computed to map P_{source} onto P_{target}. The homography matrix is given by:

$$\mathbf{H} = \mathbf{M}_{\text{target}} \mathbf{M}_{\text{source}}^{-1} \tag{4}$$

where $\mathbf{M}_{\text{source}}$ and $\mathbf{M}_{\text{target}}$ are 3×3 transformation matrices constructed from the detected and reference coordinates. This transformation is then applied to each pixel (x, y) in the original image using homogeneous coordinates:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \mathbf{H} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5}$$

where (x', y') are the transformed pixel locations after division by w'.

3.4.1 Ensuring Robust Perspective Correction

To improve the robustness of the transformation, we implement the following refinements:

- **Corner Refinement:** Since the YOLOv8 mask may introduce slight inaccuracies, we refine the detected corner points using a Harris corner detector and sub-pixel accuracy optimization.
- **Adaptive Scaling:** The transformation matrix is scaled based on the detected board size to maintain uniform piece proportions.
- **Constraint Checking:** We verify that the transformed board remains rectangular by ensuring the Euclidean distances between transformed corner points match expected values.

Once the transformation is applied, the output image contains a chessboard that is consistently aligned in a fixed reference frame. This ensures that each square in the board corresponds to a predefined position in the 8×8 grid, allowing for accurate grid segmentation and piece classification in subsequent steps.

Fig. 3: Input Image taken from an arbitrary angle

3.5 Splitting the Board into an 8×8 Grid

After applying the perspective transformation, the chessboard is aligned to a fixed top-down view, making it suitable for further processing. The next step involves segmenting the board into an 8×8 grid, where each cell corresponds to a chess square that may or may not contain a piece. This division allows for isolated classification of each individual square.

3.5.1 Grid Division Strategy

Given a transformed board image of size $W \times H$, each square is of equal dimensions, calculated as:

$$w_{\text{cell}} = \frac{W}{8}, \quad h_{\text{cell}} = \frac{H}{8} \tag{6}$$

where w_{cell} and h_{cell} denote the width and height of each individual chessboard square.

For each grid position (i, j), the pixel coordinates defining the bounding box of the square are computed as follows:

$$x_{\text{start}} = j \cdot w_{\text{cell}}, \quad x_{\text{end}} = (j+1) \cdot w_{\text{cell}} \tag{7}$$

$$y_{\text{start}} = i \cdot h_{\text{cell}}, \quad y_{\text{end}} = (i+1) \cdot h_{\text{cell}} \tag{8}$$

where $i, j \in \{0, 1, 2, \ldots, 7\}$ represent the row and column indices, respectively.

Each extracted square, denoted as $S_{i,j}$, is then obtained from the image I using:

$$S_{i,j} = I[y_{\text{start}} : y_{\text{end}}, x_{\text{start}} : x_{\text{end}}] \tag{9}$$

This operation efficiently extracts the individual chessboard squares in a structured manner, ensuring that each region is properly cropped for further analysis.

3.5.2 Handling Board Resizing and Scaling

Due to variations in camera resolution and chessboard dimensions, it is essential to normalize the extracted grid squares. A standard target resolution, such as 64×64 pixels per square, is used to resize each cell while preserving aspect ratios:

$$S_{i,j} \leftarrow \text{resize}(S_{i,j}, 64, 64) \tag{10}$$

where the resize$(\cdot)$ operation ensures that all extracted regions conform to the input dimensions required by the classification model.

3.5.3 Challenges and Considerations

Several challenges arise in the board-splitting process:

- **Misaligned Boards:** If the perspective transformation is not perfectly accurate, some squares may be slightly distorted. This is corrected by refining the homography matrix using sub-pixel corner adjustments.
- **Variable Lighting Conditions:** Uneven lighting across the board can cause inconsistent contrast between squares. To mitigate this, adaptive histogram equalization is applied to normalize brightness levels.
- **Chess Piece Overlaps:** Some chess pieces may extend beyond their designated squares, partially overlapping neighboring regions. A margin-based segmentation approach is employed to account for this.

Each extracted square is then passed to the YOLOv11-based chess piece classification model, which assigns a label indicating whether the square is empty or occupied by a specific chess piece. The classified grid is subsequently converted into FEN notation to represent the chessboard state.

3.6 Chess Piece Classification using YOLOv11

Once the chessboard has been segmented into 64 squares, each square is analyzed to determine whether it contains a chess piece and, if so, which piece it is. To achieve this, we trained a YOLOv11-based classification model to recognize the different chess pieces. The model assigns each extracted square to one of 13 possible classes, which include:

- **Six White Pieces:** King (K), Queen (Q), Rook (R), Bishop (B), Knight (N), Pawn (P).

- **Six Black Pieces:** King (k), Queen (q), Rook (r), Bishop (b), Knight (n), Pawn (p).
- **Empty Square:** Represented by a special class ($\varnothing$).

3.6.1 Model Architecture and Output Representation

YOLOv11 is a state-of-the-art deep learning model designed for real-time object classification. Given an input image patch $S_{i,j}$ from the chessboard, the model produces a probability distribution $\mathbf{P}$ over the 13 possible classes:

$$\mathbf{P} = \{p_0, p_1, \ldots, p_{12}\}, \quad \sum_{i=0}^{12} p_i = 1 \tag{11}$$

where each probability value p_i represents the likelihood of the corresponding class. The final predicted class for a given square is determined using the argmax operation:

$$c_{\text{pred}} = \arg\max_i p_i \tag{12}$$

where c_{pred} is the class index with the highest probability.

3.6.2 Inference and Mapping to Chessboard State

After classification, each detected piece is assigned to its corresponding board position (i, j). The final chessboard configuration is represented as an 8×8 matrix:

$$B = \{c_{\text{pred}}(i, j) \mid 0 \leq i, j \leq 7\} \tag{13}$$

where B contains the predicted labels for all 64 squares. This matrix is then converted into FEN (Forsyth-Edwards Notation), a standard representation of chessboard states, using:

$$\text{FEN} = \text{convert\_to\_FEN}(B) \tag{14}$$

This notation is used to store, analyze, and further process the board state for chess engine compatibility.

For the given input image shown in Figure **??**, the obtained FEN representation is:

$$8/q2NP2r/2b5/1n3B2/R2P4/3k1p2/1K5Q/8\ w\ -\ -\ 0\ 1$$

When this FEN string is input into a chess analysis platform, the corresponding board position is accurately reconstructed, as illustrated in Figure 3. This demonstrates the effectiveness of the proposed system in generating precise FEN notations for real-world chessboard images.

3.6.3 Challenges and Considerations

- **Occlusions and Partial Visibility:** Some pieces may partially obscure others, especially in real-world scenarios. To address this, we employed data augmentation with synthetic occlusions during training.

Fig. 4: Wrapped Chessboard Input Image with the Classification Output Labels

- **Class Imbalance:** Since certain pieces (e.g., pawns) appear more frequently than others (e.g., kings), we balanced the dataset using class-weighted loss functions.
- **False Positives in Empty Squares:** The model was fine-tuned to reduce misclassification of empty squares by introducing background noise variations in the dataset.

With these enhancements, the YOLOv11 model provides accurate and efficient classification of chess pieces, enabling seamless FEN generation and further chess analysis.

3.7 Converting to FEN Notation

Forsyth-Edwards Notation (FEN) is a widely used standard for representing chessboard states in a single line of text. It encodes the position of all pieces, the active player's turn, castling rights, en passant possibilities, and move counters. This representation allows easy integration with chess engines and analysis tools.

3.7.1 FEN Format

A FEN string consists of six space-separated fields:

$$\text{FEN} = \underbrace{\text{row}_1/\text{row}_2/\ldots/\text{row}_8}_{\text{Piece Placement}} \quad \underbrace{\text{turn}}_{\text{Active Player}} \quad \underbrace{\text{castling}}_{\text{Castling Rights}} \quad \underbrace{\text{en passant}}_{\text{En Passant Target}} \quad \underbrace{\text{halfmove}}_{\text{Halfmove Clock}} \quad \underbrace{\text{fullmove}}_{\text{Fullmove Counter}}$$

where:

- **Row Representations:** The board is described row by row from rank 8 to rank 1. Each row contains piece symbols and empty square counts.
- **Turn:** Either 'w' (White to move) or 'b' (Black to move).
- **Castling Rights:** A string indicating castling availability ('KQkq' for White king/queen-side, 'kq' for Black).

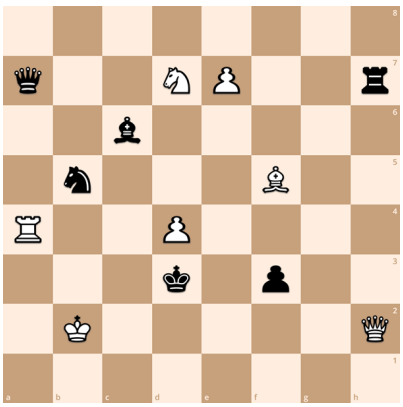

Fig. 5: Converted Input 3-D Chessboard into 2-D Board with the Obtained FEN Score

- **En Passant Target:** If a pawn has just moved two squares forward, the target square is noted (e.g., ‘e3‘); otherwise, ‘-‘.
- **Halfmove Clock:** The number of halfmoves since the last capture or pawn move, used for the fifty-move rule.
- **Fullmove Counter:** Increments after each Black move, starting from 1.

3.7.2 Generating FEN from Classified Board State

Once the chessboard has been classified into an 8×8 matrix:

$$B = \{c_{\text{pred}}(i, j) \mid 0 \leq i, j \leq 7\} \tag{15}$$

the FEN string is constructed row-wise. Each row is converted as follows:

- A sequence of chess piece symbols (‘K‘, ‘Q‘, ‘R‘, ‘B‘, ‘N‘, ‘P‘ for White; lowercase for Black).
- Consecutive empty squares are replaced by a single digit (e.g., ‘8‘ for a full empty row).
- Rows are separated by the ‘/‘ character.

The generated row representations are concatenated to form the first field of the FEN string.

3.7.3 Example: Initial Chess Position

For the standard starting position:

$$B = \begin{bmatrix} r & n & b & q & k & b & n & r \\ p & p & p & p & p & p & p & p \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ P & P & P & P & P & P & P & P \\ R & N & B & Q & K & B & N & R \end{bmatrix} \tag{16}$$

The corresponding FEN string is given by:

"rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1"

where:

- **Piece Placement:** `rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR` represents the arrangement of pieces on the board, with lowercase letters denoting black pieces and uppercase letters denoting white pieces.
- **Turn Indicator:** `w` specifies that it is White's turn to move.
- **Castling Rights:** `KQkq` indicates that all castling rights are available:
  - `K` - White can castle kingside.
  - `Q` - White can castle queenside.
  - `k` - Black can castle kingside.
  - `q` - Black can castle queenside.
- **En Passant Target:** `-` denotes that there is no en passant target square.
- **Halfmove Clock:** `0` indicates that no pawn moves or captures have been made since the last irreversible move.
- **Fullmove Number:** `1` represents the current move number in the game (starting from 1).

3.7.4 Integrating FEN into Chess Engines

The generated FEN strings ensure compatibility with chess engines such as Stockfish and Leela Chess Zero (LCZero). These engines utilize FEN to validate board configurations, ensuring that detected positions adhere to chess rules. They can analyze the given position and suggest optimal moves, enhancing decision-making in real time. Additionally, FEN strings assist in evaluating the accuracy of the recognition system by comparing predicted positions with ground truth data. This facilitates debugging and model refinement, making FEN a crucial component for integrating automated board recognition with advanced chess analysis tools.

Thus, the FEN conversion step is a crucial bridge between computer vision-based chessboard recognition and practical chess applications.

4 Results and Discussions

4.1 Comparative Analysis: CNN vs. YOLOv11 Cls

A comparative evaluation between the CNN-based classifier and the YOLOv11 classification model was conducted, focusing on their performance in chess piece recognition. Figures 6 and 8 illustrate the confusion matrices of CNN and YOLOv11 Cls, respectively, while Figures 7 and 9 depict their predicted labels.

4.1.1 CNN Performance and Limitations

The CNN model achieved an accuracy of **72.7%**, with notable misclassifications, particularly between visually similar pieces such as pawns and bishops. This can be attributed to the limited ability of CNNs to retain spatial relationships, leading to confusion in subtle structural differences. Furthermore, the model performed poorly in challenging conditions, such as low lighting and occlusions, where shadows and contrast variations affected feature extraction.

CNN operates by extracting spatial hierarchies of features using convolutional layers, followed by pooling layers to reduce dimensionality. However, due to its fixed receptive fields, it struggles with variations in scale, rotation, and illumination. As seen in Figure 7, CNN misclassified several pieces under these conditions, demonstrating its limitations in real-world scenarios. The confusion matrix of the best-performing fold from 5-fold cross-validation (Figure 6) highlights these classification difficulties.

4.1.2 Why YOLOv11 Cls?

In contrast, YOLOv11 Cls achieved a significantly higher accuracy of **95.4%**, demonstrating superior performance in distinguishing between chess pieces. Unlike CNN, which processes fixed-size patches, YOLOv11 leverages an anchor-based detection mechanism alongside a classification head, maintaining spatial relationships across the entire image. This allows the model to better recognize pieces even under different lighting conditions, board orientations, and slight occlusions.

As observed in Figure 8, the confusion matrix of YOLOv11 Cls shows minimal misclassifications compared to CNN. The model effectively adapts to varying conditions, resulting in fewer errors (Figure 9). Additionally, its real-time inference capability makes it a more practical choice for chessboard analysis in dynamic environments.

Overall, the YOLOv11 Cls model outperformed CNN significantly, offering a more reliable and accurate classification approach. Its ability to preserve spatial context and distinguish fine-grained differences between pieces makes it the preferred choice for chessboard recognition.

4.2 FEN Generation Process

The process of generating the FEN (Forsyth-Edwards Notation) score from a chessboard image involves multiple steps, as depicted in Figures 10, 11, and 12.

1. **Input Image Acquisition (Figure 10)**:
   - The input image is captured from an arbitrary angle.

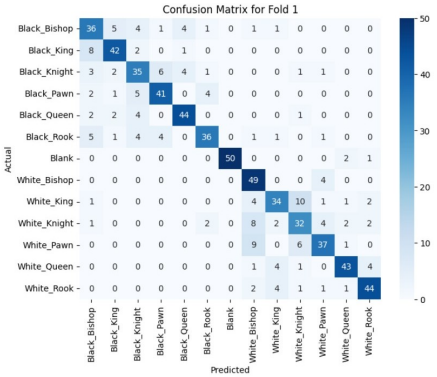

Fig. 6: Confusion Matrix of CNN

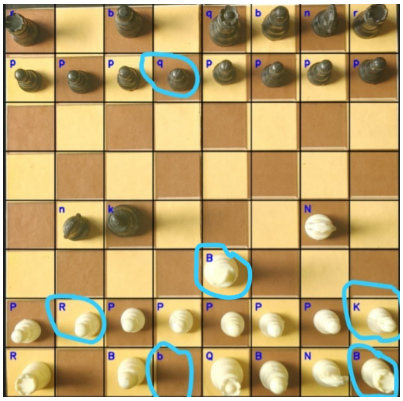

Fig. 7: Predicted Lables by CNN Model showing many misclassifications

- Variations in lighting, perspective distortion, and background noise may be present.
- The raw image undergoes preprocessing to correct perspective and extract the chessboard region using YOLOv8.

2. **Chessboard Wrapping and Piece Classification (Figure 11)**:
   - A perspective transformation is applied to convert the 3D-viewed chessboard into a top-down 2D representation.
   - The board is divided into 64 squares, and each square is individually processed.
   - YOLOv11 then classifies each square as either a blank space or a chess piece.
   - The detected pieces are labeled accordingly, producing an annotated board representation.
3. **FEN Generation (Figure ??)**:
   - The classified board is converted into a structured 8×8 matrix.

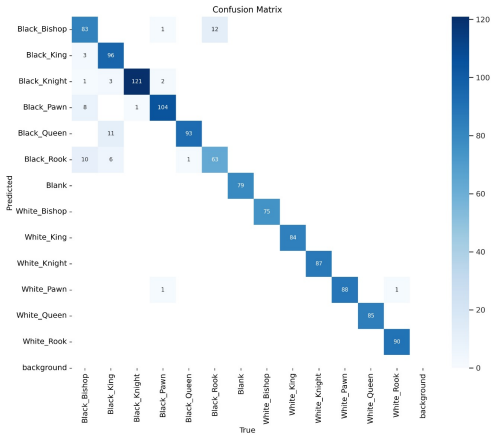

Fig. 8: Confusion Matrix of YOLOv11 Cls

Fig. 9: Predicted Lables by YOLOv11 Clas accurate classifications

- Each piece label is mapped to its corresponding FEN notation.
- The final FEN string is generated by following standard notation rules, including piece arrangement, active color, castling rights, en passant targets, and move counters.
- This FEN string can be used for chess game analysis, validation, or AI-based move predictions.

5 Conclusion and Future Work

This paper presents an end-to-end pipeline for real-time chessboard recognition and Forsyth-Edwards Notation (FEN) conversion using deep learning techniques. Our methodology integrates YOLOv8-based segmentation for board detection, perspective correction for alignment, an 8×8 grid-splitting algorithm, and a YOLOv11-based

Fig. 10: Input Image taken from an arbitrary angle

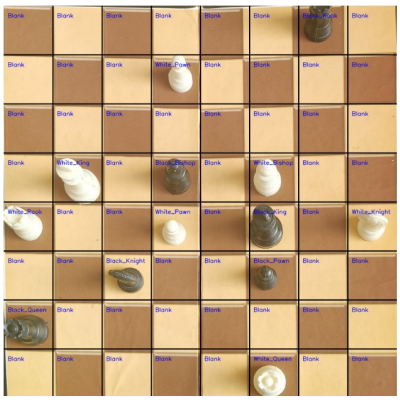

Fig. 11: Wrapped Chessboard Input Image with the Classification Output Labels

classification model for piece recognition. The extracted board state is then converted into FEN notation, enabling seamless integration with chess engines and game analysis tools. Experimental results demonstrate that our model achieves high accuracy in both board detection and piece classification under varied real-world conditions, making it suitable for practical applications such as automated game recording and real-time chess analysis.

For future work, we aim to improve classification accuracy by incorporating transformer-based vision models for enhanced spatial feature learning. Additionally, we plan to introduce a multi-frame temporal consistency approach, ensuring stability in classification across video streams. Another promising direction is the development of a self-supervised learning framework, reducing the dependency on large annotated datasets. Furthermore, extending our system to recognize handwritten chess notations and support 3D chessboard perspectives would broaden its applicability. Finally,

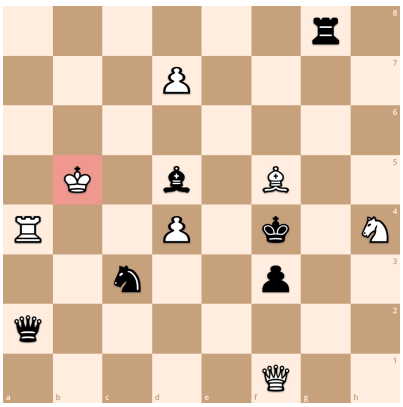

Fig. 12: Converted Input 3-D Chessboard into 2-D Board with the Obtained FEN Score

Fig. 13: Input Image taken from an arbitrary angle

deploying the model on edge devices such as mobile phones or embedded systems would enhance accessibility for real-time chess enthusiasts.

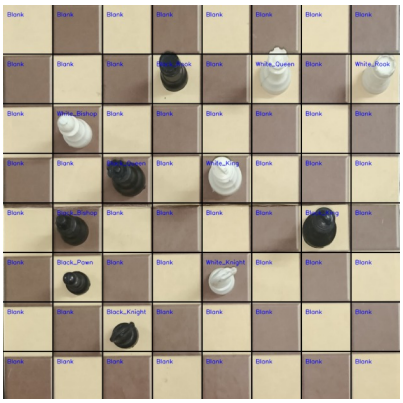

Fig. 14: Wrapped Chessboard Input Image with the Classification Output Labels

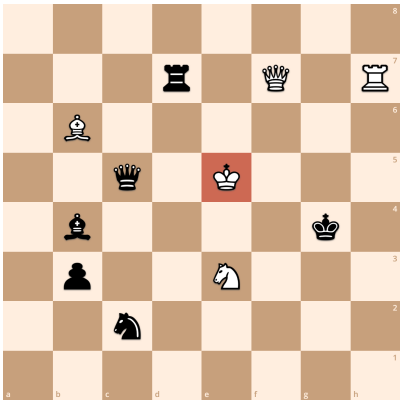

Fig. 15: Converted Input 3-D Chessboard into 2-D Board with the Obtained FEN Score

Fig. 16: Input Image taken from an arbitrary angle

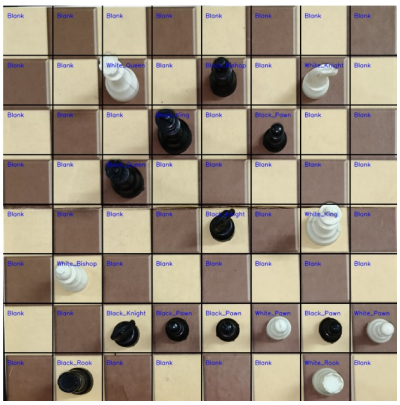

Fig. 17: Wrapped Chessboard Input Image with the Classification Output Labels

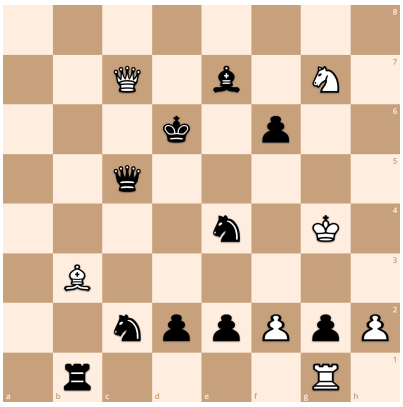

Fig. 18: Converted Input 3-D Chessboard into 2-D Board with the Obtained FEN Score

Fig. 19: Input Image taken from an arbitrary angle

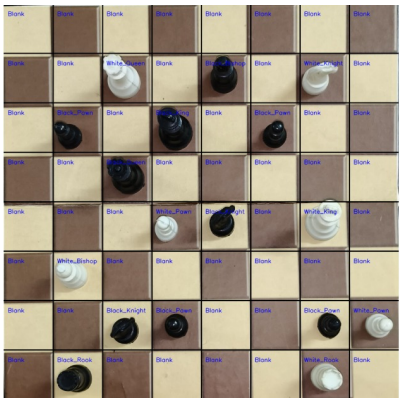

Fig. 20: Wrapped Chessboard Input Image with the Classification Output Labels

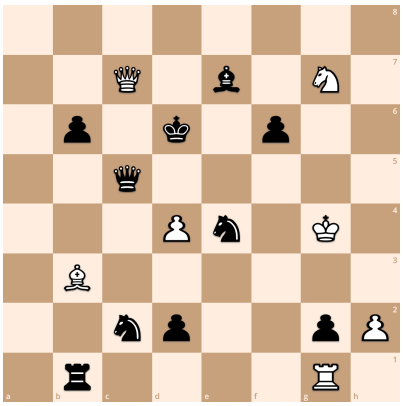

Fig. 21: Converted Input 3-D Chessboard into 2-D Board with the Obtained FEN Score

Fig. 22: Input Image taken from an arbitrary angle

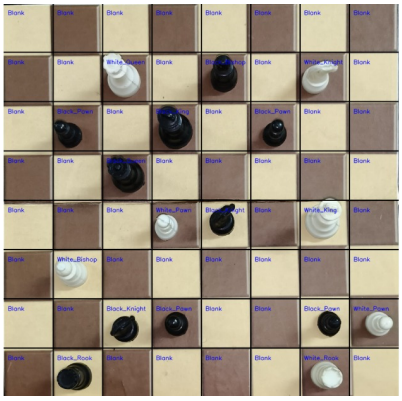

Fig. 23: Wrapped Chessboard Input Image with the Classification Output Labels

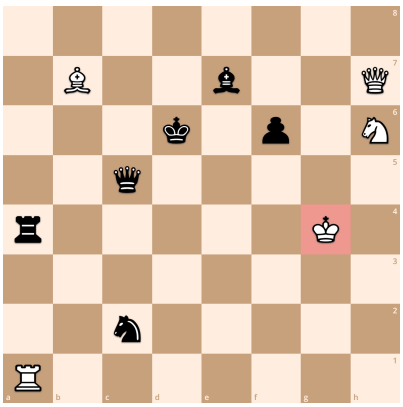

Fig. 24: Converted Input 3-D Chessboard into 2-D Board with the Obtained FEN Score

4.3 Failure Analysis and Potential Solutions

Reasons for Failure:

- **Similar Appearance from Top View:** Pieces such as bishops and pawns have similar top profiles, leading to misclassification.
- **Lighting Variations:** Uneven or dark lighting conditions create inconsistent shadows, affecting model predictions.
- **Piece Overlapping:** In some cases, pieces may be incorrectly labeled as multiple pieces due to overlapping predictions.
- **Board Misalignment:** If the board is slightly rotated or distorted, the grid-based detection may misinterpret positions.

Potential Solutions:

- **Enhanced Data Augmentation:** Introduce synthetic lighting variations and minor rotations to make the model more robust.
- **Multi-Angle Training:** Train the model with images captured from slightly different perspectives to differentiate similar pieces.
- **Context-Aware Corrections:** Implement a chess-rule-based post-processing step to validate FEN outputs and correct inconsistencies.
- **Refined Segmentation:** Improve YOLO segmentation by using higher-resolution training data to avoid overlapping piece detection.
- **Adaptive Thresholding:** Use adaptive contrast enhancement to reduce the impact of shadows and uneven lighting.

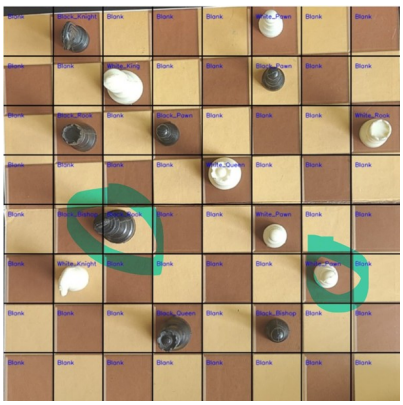

Fig. 25: Cases where the Prediction fails

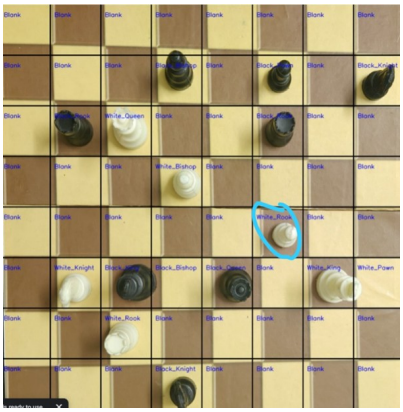

Fig. 26: Cases where the Prediction fails