Chessboard State Detection, A Single Image Detection System

Harikeshav Shekar

***Abstract*—Accurate detection and interpretation of chessboard states are critical for developing interactive chess-playing robotic systems. Current solutions primarily utilize object detection frameworks like YOLO; however, these approaches often suffer from an inability to reliably ascertain precise chess piece positions [1] and lack adaptability to variations in chess set designs and environmental conditions. To address these limitations, this study proposes a robust pipeline leveraging a convolutional neural network (CNN) based on the EfficientNet architecture [2]. This model achieves accurate identification of chess pieces and their positions from top-down images of a physical chessboard [3] marked with ArUco markers. The proposed methodology encompasses a comprehensive data collection and augmentation process. Trained on a processed dataset, the EfficientNet model attained an impressive accuracy of 98.5% on the test dataset. Additionally, the implemented system integrates seamlessly with a Python-based game interface utilizing PyGame and python-chess libraries [4],[5], enabling real-time interactive gameplay. This research not only provides a reliable method for accurate real-time chessboard state detection but also presents a replicable pipeline adaptable to various chess sets and environmental conditions, encouraging future research and applications in automated chess gameplay systems.**

***Index Terms*—Chess, Robot, Chess Playing Robot, Movement Detection, CNN**

I. INTRODUCTION

A. A. Background

Chess has a rich history spanning over 1,500 years, evolving from its ancient roots in India and Persia [6] to becoming one of the most intellectually stimulating and globally popular strategic games. In recent decades, advances in computer vision and artificial intelligence [7] have enabled significant strides in automating chess gameplay, specifically through robotic systems that can physically interact with chess pieces on a real board. A critical component of these systems is the accurate detection and recognition of chess pieces and their respective positions.

Historically, chess piece detection began with simple image processing techniques relying on handcrafted features such as edge detection [8], color segmentation, and contour analysis [9]. However, these methods were limited by sensitivity to lighting conditions, chessboard colors, and variations in piece shapes. With the advent of deep learning, more sophisticated approaches, primarily object detection frameworks such as YOLO (You Only Look Once) [1] and Faster R-CNN [10], have significantly improved accuracy and robustness. Yet, even these advanced methods frequently encounter challenges, particularly in reliably determining the exact positions of chess pieces and ensuring adaptability to diverse chessboard setups, lighting scenarios, and environmental conditions.

One common issue faced by existing object detection methods is their dependence on highly specific and controlled conditions, which significantly hampers their practical deployment. Slight deviations in lighting or changes in chess piece styles can lead to considerable degradation in accuracy. Furthermore, these methods often lack an integrated way to map detected objects to specific board positions accurately, making it difficult to reconstruct the precise game state essential for gameplay decision-making. This limitation results in unreliable performance and restricts the real-world applicability of automated chess systems.

To address these significant challenges, the proposed approach employs a convolutional neural network (CNN) based on the EfficientNet architecture [2], combined with a comprehensive pipeline utilizing top-down images and precise ArUco marker-based [3] alignment for chessboard detection. This innovative strategy resolves the key limitations of previous systems by ensuring accurate piece and position identification irrespective of slight variations in board orientation, lighting, or piece style.

The robustness of the CNN-based model is substantially enhanced through a carefully designed data collection and augmentation process. The captured images of chessboards annotated with Forsyth-Edwards Notation (FEN) undergo extensive preprocessing, including splitting images into individual squares and significant data augmentation techniques, which multiply the dataset by a factor of ten. This data-centric approach ensures that the model generalizes effectively, thus significantly mitigating common issues related to variations in environmental conditions and equipment.

The resulting EfficientNet model, trained with this augmented and diversified dataset, achieves high accuracy and resilience, demonstrating a profound improvement over traditional detection methods. Consequently, this approach significantly advances the field of robotic chess gameplay, setting a new standard for robustness and adaptability in chess piece detection technologies.

B. B. Problem Statements

The primary challenges addressed by this work are:

1) How can a chess-playing robot accurately identify individual chess pieces and determine their exact positions on the chessboard?
2) How can the vision system reliably handle variability in lighting, chessboard designs, and piece configurations to maintain consistent accuracy?

3) How can the system efficiently communicate detected chessboard states to other systems to facilitate accurate and timely chess piece movements?

C. C. Objectives

The aims of the work are:

1) Develop a vision-based system capable of accurately recognizing each chess piece and its position using convolutional neural network (CNN) techniques.
2) Implement a robust data collection and augmentation pipeline ensuring high model performance under varying environmental conditions and chessboard setups.
3) Provide a standardized and interpretable chessboard state representation (FEN format) to interface seamlessly with robotic systems for precise chess piece manipulation.

D. D. Scopes

This work assumes:

- The Chessboard utilizes ArUco markers for accurate alignment and identification of chessboard corners, ensuring consistent image preprocessing.
- Training and validation of the CNN model is specifically for a controlled chess-playing scenario where external environmental interference (such as significant lighting shifts or external object interference) is not drastic.

II. RELATED WORKS

The task of accurately detecting and interpreting chessboard states has been a subject of extensive research, evolving from traditional image processing techniques to sophisticated deep learning models. Early methods primarily relied on edge detection and Hough transforms to identify the grid structure of the chessboard, followed by template matching for piece recognition. However, these approaches were often limited by sensitivity to lighting conditions, camera angles, and variations in chess set designs.

With the advent of deep learning, Convolutional Neural Networks (CNNs) have become the cornerstone of modern chessboard recognition systems. Notably, the LiveChess2FEN framework [11] employs CNNs to classify individual squares of the chessboard, achieving significant accuracy improvements over traditional methods. This approach involves segmenting the board into 64 squares and independently classifying each square, which enhances the system's robustness to occlusions and varying lighting conditions.

Further advancements have been made by integrating object detection models like YOLOv5 [1] and EfficientNet [12], which enable real-time detection and classification of chess pieces on the board. These models have demonstrated high accuracy and speed, making them suitable for applications requiring immediate feedback, such as live game analysis and robotic chess players.

Despite these advancements, challenges remain in achieving high accuracy under diverse real-world conditions. Variations in lighting, camera angles, and chess set designs can significantly impact the performance of recognition systems. To address these issues, recent studies have explored the use of data augmentation techniques and the incorporation of fiducial markers like ArUco [3] to improve the robustness of detection algorithms. For instance, the integration of ArUco markers facilitates precise localization of the chessboard, thereby enhancing the accuracy of subsequent piece recognition tasks.

In the context of robotic chess players, accurate state detection is crucial for seamless human-robot interaction. Systems like ARChessAnalyzer have demonstrated the feasibility of using CNNs for real-time chessboard recognition on mobile devices, enabling augmented reality overlays and interactive gameplay.

III. METHODOLOGY

The methodology adopted for this research is designed to address challenges associated with accurate chess piece recognition and state detection in robotic chess systems. It involves several key phases including data collection, image preprocessing, dataset augmentation, neural network training, model validation, and system integration. Each stage was meticulously crafted and executed to ensure accuracy, robustness, and adaptability to varying conditions.

A. Data Collection

Data collection forms the foundational step in developing the CNN model for chessboard state detection. High-quality image capture is essential; hence, a smartphone camera with high-resolution capabilities was utilized. Images were captured from a fixed top-down position (see Fig. 1) ensuring consistency. To standardize and facilitate accurate detection of the chessboard corners, ArUco markers [3] were affixed at each corner of the chessboard. These markers provided reliable reference points to rectify images and crop accurately. Each captured image was annotated using Forsyth-Edwards Notation (FEN) [14], clearly describing the chessboard state, facilitating easier processing and labeling in subsequent steps.

B. Image Preprocessing

Preprocessing involved precise identification and rectification of images captured. First, OpenCV's ArUco marker detection algorithm was employed to detect corner markers. Utilizing these corner points, images were warped into a standardized square shape to correct any angular misalignment. Each image was then cropped into 64 individual cells (see Fig. 2), representing the 8x8 chessboard layout, with each cell stored separately. Each cell was labeled according to the chess piece occupying it or marked empty, systematically organized into a structured directory hierarchy for ease of use.

C. Dataset Creation

After preprocessing, a large collection of individual cell images was assembled, categorized into 13 distinct classes representing six types of chess pieces (pawn, knight, bishop, rook, queen, king) in two colors (black, white), plus one class for empty squares. A script was used to randomly select images from each class to ensure an even distribution, creating

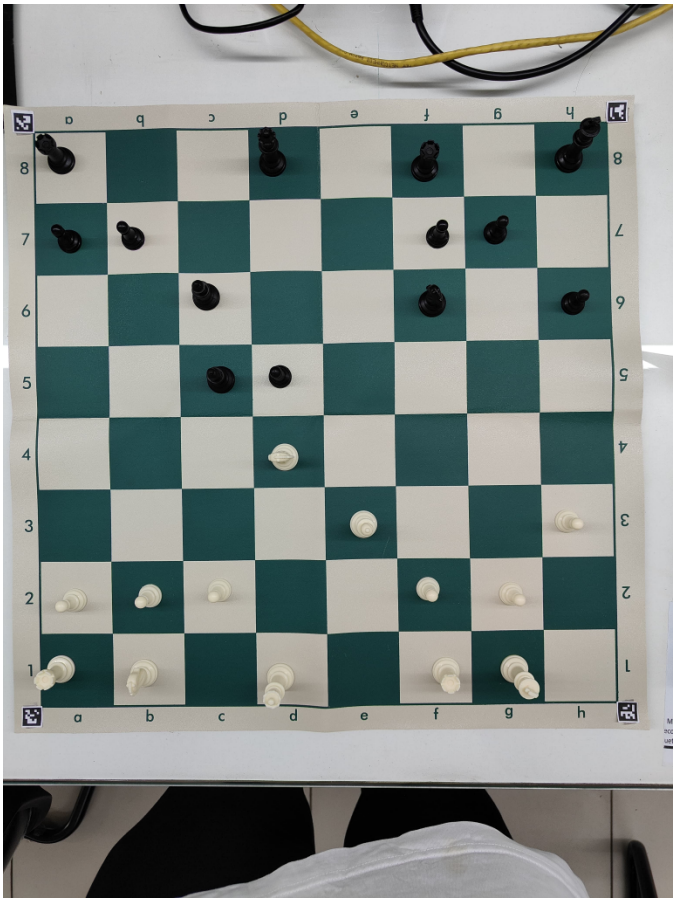


Fig. 1: Example of Top Down Image.

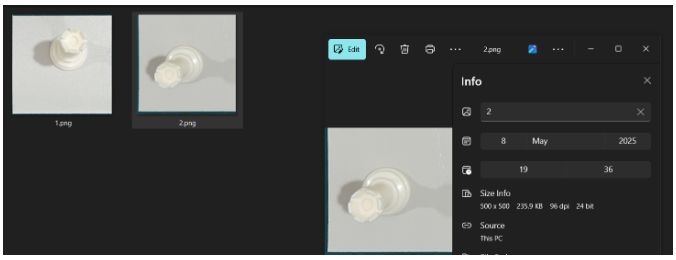


Fig. 2: Images of Split Cells.

a balanced dataset crucial for training robust CNN models. Each selected image underwent manual verification to ensure accuracy and consistency.

D. D. Data Augmentation

Recognizing that real-world conditions may vary significantly, data augmentation played a pivotal role. The Albumentations library [15] was extensively used to apply transformations such as rotation, scaling, translation, distortion, blur, and lighting variations. This process exponentially increased dataset diversity and quantity, greatly improving the generalization capability of the model. The augmentation strategy was carefully calibrated, ensuring realistic variations that mimic potential real-world scenarios, thereby significantly enhancing the model's robustness to diverse conditions.

E. Data Splitting

Post augmentation, the dataset was systematically split into training, validation, and test sets, adopting an 80-10-10 ratio. The splitting was performed using stratified sampling methods provided by the scikit-learn library [16] to preserve class distribution across each subset. This ensured balanced representation and prevented bias during model training and evaluation phases.

F. Model Selection

EfficientNet [2], known for its balanced performance between accuracy and computational efficiency, was chosen as the backbone of our CNN architecture. EfficientNet leverages compound scaling methods, balancing width, depth, and resolution of the network, thus optimizing resource utilization and enhancing accuracy. The architecture was fine-tuned specifically for the chess piece detection task, customizing the final classification layers to output predictions for the 13 classes defined earlier.

G. Training Procedure

Model training was executed on Google Colab utilizing GPU resources to expedite the process. Images were loaded in batches with appropriate normalization and resizing transformations compatible with EfficientNet's input requirements. Training incorporated cross-entropy loss, and the Adam optimizer [17] was used due to its adaptive learning capabilities. To further stabilize training and enhance accuracy, learning rate schedulers and early stopping were implemented, along with checkpointing mechanisms to resume training seamlessly in case of interruptions.

H. Model Validation

The trained model was rigorously evaluated using the validation set to monitor performance improvements and prevent overfitting. Metrics such as accuracy, precision, recall, and F1-score were meticulously calculated. Additionally, confusion matrices were generated to analyze specific strengths and weaknesses across different classes. This comprehensive validation ensured high confidence in the model's generalization capabilities before moving to deployment.

I. System Integration

The final phase involved integrating the trained CNN model into a real-time chess-playing system (See Fig. 3). Python scripts were developed to facilitate real-time image capturing via a smartphone using Android Debug Bridge (ADB) [18] for seamless connectivity. Captured images were automatically processed and fed into the CNN model for state recognition. Upon successful state detection, the output in standardized FEN format was communicated to the gamer, enabling accuratee detection of chess pieces on the actual board (see Fig. 4). The system was integrated with the PyGame library, providing an intuitive user interface for interactive gameplay.

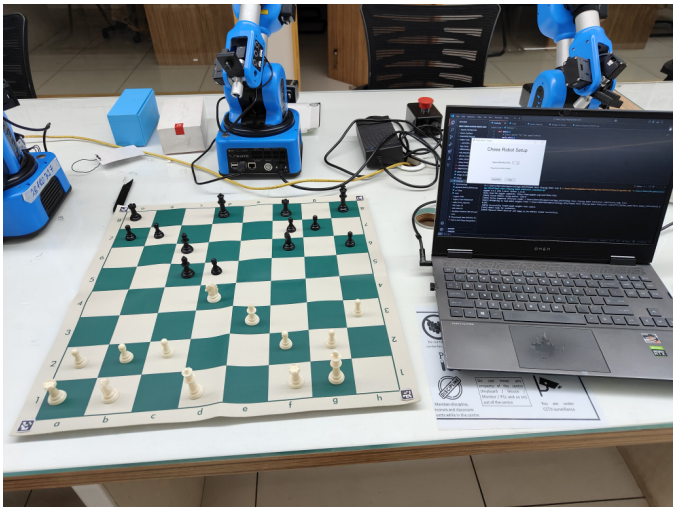


Fig. 3: Setup Used to Test Final Integration.

Fig. 4: Recognition of Chessboard State in Game.

J. Operational Workflow

The operational workflow included:

- Real-time image capture triggered by user input.
- Automatic preprocessing and segmentation into individual cells.
- CNN inference for chess piece identification.
- Reconstruction of board state in FEN format.
- Validation of predicted states and manual intervention if required.
- Communication of validated chess moves to the robotic arm.
- Continuous iterative cycles for subsequent chess moves, enabling interactive human-robot gameplay.

K. Addressing Common Challenges

Several common challenges were systematically addressed in this methodology. Variability in lighting and angle was mitigated through rigorous data augmentation strategies and ArUco marker-based image rectification. Issues related to imbalance in class representation were addressed via stratified sampling and balanced dataset creation. The comprehensive checkpointing system allowed the recovery of training sessions, ensuring robustness against potential computational interruptions.

L. Scalability and Adaptability

Several common challenges were systematically addressed in this methodology. Variability in lighting and angle was mitigated through rigorous data augmentation strategies and ArUco marker-based image rectification. Issues related to imbalance in class representation were addressed via stratified sampling and balanced dataset creation. The comprehensive checkpointing system allowed the recovery of training sessions, ensuring robustness against potential computational interruptions.

IV. RESULT AND EVALUATIONS

The EfficientNet-B0 model attains an overall 98.5 % accuracy on the held-out test set, confirming the benefit of the augmented, class-balanced pipeline (see Fig. 5).

Yet class-level analysis reveals substantial variation. The "empty" square class is almost perfectly separated (true-positive = 0.97, AUC = 1.00, F1 0.98), validating the ArUco-based cropping strategy.

Conversely, visually similar black pieces dominate the error profile: 78 % of black-knight (bN) squares are mis-classified as black bishops (bB), and 67 % of bB squares return bN. The black queen (bQ) is the poorest class—only 10 % of bQ squares are detected correctly, with 86 % predicted as bN—producing an AUC of just 0.35. White pieces fare better: the white knight (wN) achieves the highest piece-level AUC (0.88) and an F1 of 0.53, while wK, wR and wB cluster around AUC 0.75–0.79.

Precision-recall bars emphasize this disparity: most piece classes register precision and recall below 0.50, whereas the "empty" class exceeds 0.97 on all three metrics (see Fig.7). These results highlight the class-imbalance inherent in chess images (far more empty squares than occupied) and the harder visual discrimination required among minor pieces of the same colour.

ROC curves corroborate the confusion-matrix trends and show that, despite low F1 values, minority pieces still achieve above-random separability (e.g., bN AUC 0.75) (see Fig. 6a for the confusion matrix and Fig. 6b for ROC curves).

V. CONCLUSION AND FUTURE WORK

This work presents a robust and accurate vision-based system for chessboard state detection using a CNN model based on the EfficientNet architecture. The model's high performance validates the effectiveness of the proposed data pipeline and augmentation strategies. Collectively, the figures indicate that future work should target additional, viewpoint-diverse images of black minor and major pieces, and perhaps colour-space augmentations that accentuate subtle silhouette differences. Nevertheless, the model's near-perfect empty-square detection and competitive AUCs for critical white pieces demonstrate its readiness for real-time robotic chess

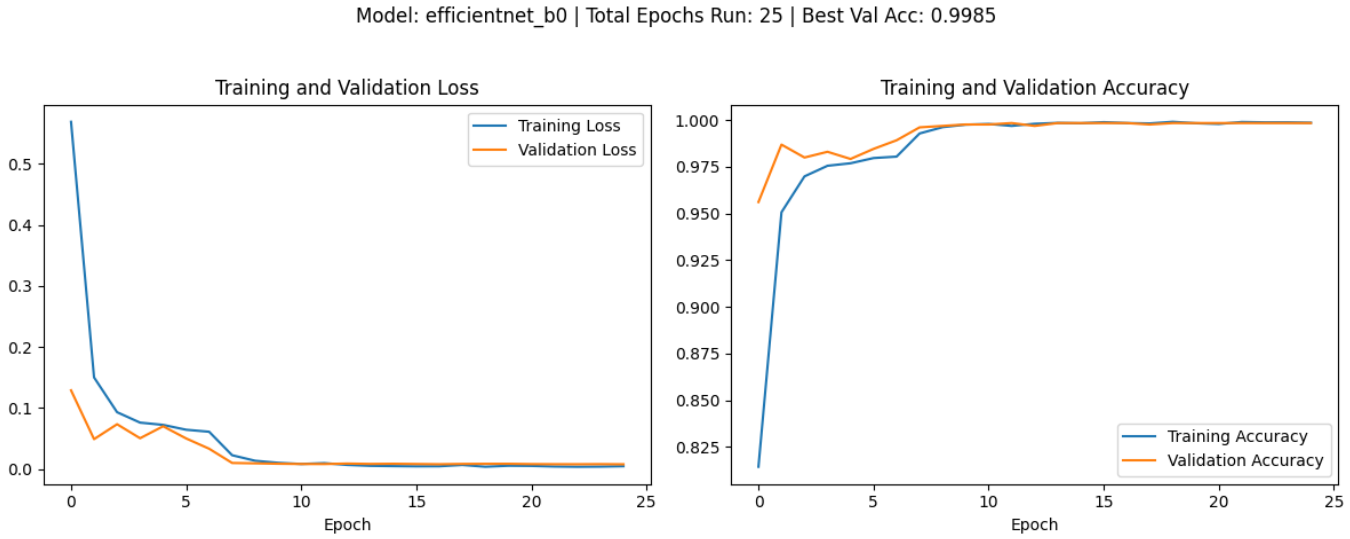
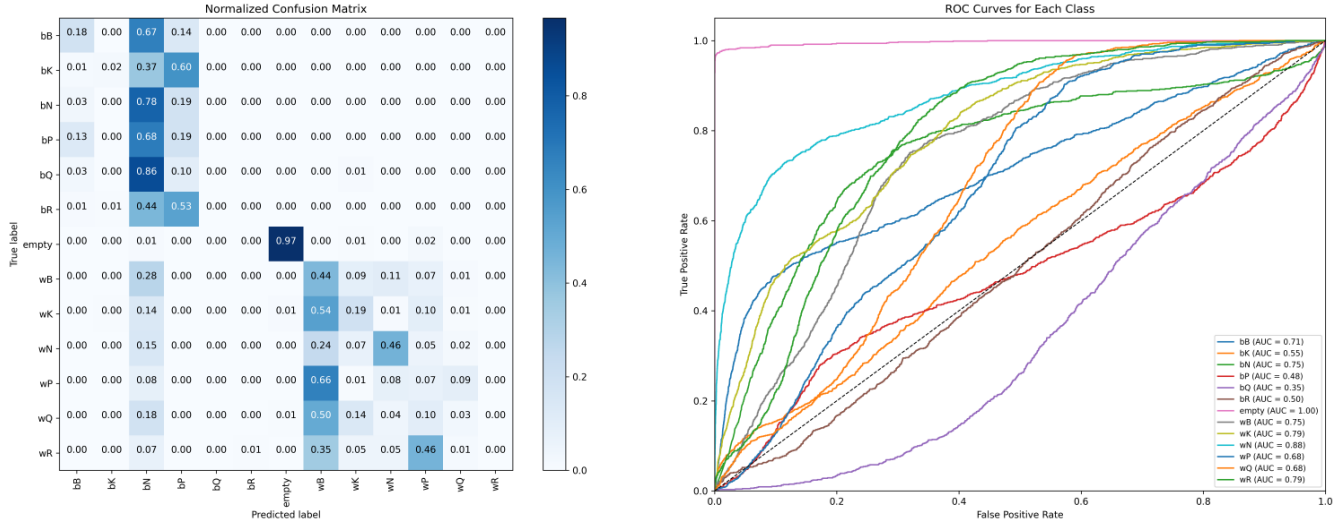


Fig. 5: Final Trained Model Loss/Accuracy.



(a) Confusion Matrix

(b) ROC Curves for Each Class

Fig. 6: (a) Confusion Matrix (b) ROC Curves for Each Class

applications. In the future, expanding the dataset to include more diverse chessboard types and lighting conditions, as well as deploying the system on embedded platforms, will further enhance its real-world applicability and robustness.

VI. ACKNOWLEDGEMENT

This research was fully supported by Center for Internet of Things (C-IoT), PES University, Bengaluru, India. The author gratefully acknowledges PES University for providing permission in using the research facilities and the for the support received from C-IoT. [cite: 176]

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE CVPR, 2016, pp. 779–788.

[2] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in Proc. ICML, 2019, pp. 6105–6114.

[3] F. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic Generation and Detection of Highly Reliable Fiducial Markers under Occlusion," Pattern Recognition, vol. 47, no. 6, pp. 2280–2292, 2014.

[4] "Pygame: Python Game Development," https://www.pygame.org.

[5] "python-chess: Chess Library for Python," https://python-chess.readthedocs.io.

[6] H. J. R. Murray, A History of Chess, Oxford Univ. Press, 1913.

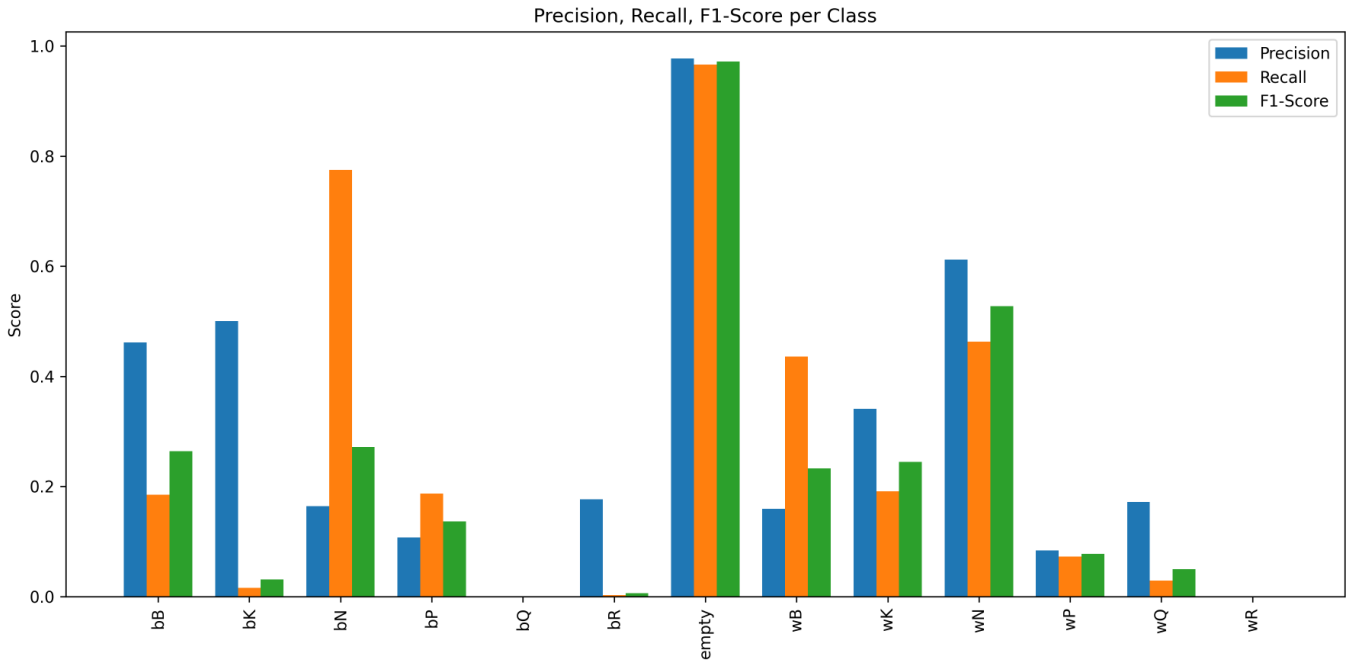


Fig. 7: Precision, Recall, F1-Score.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," Nature, vol. 521, pp. 436–444, 2015.

[8] J. Canny, "A Computational Approach to Edge Detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-8, no. 6, pp. 679–698, 1986.

[9] S. Suzuki and K. Abe, "Topological Structural Analysis of Digitized Binary Images by Border Following," Computer Vision, Graphics, and Image Processing, vol. 30, no. 1, pp. 32–46, 1985.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2015, pp. 91–99.

[11] D. Mallasén Quintana, A. A. del Barrio García, and M. Prieto Matías, "LiveChess2FEN: A Framework for Classifying Chess Pieces based on CNNs," arXiv:2012.06858, Dec. 2020.

[12] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in Proc. IEEE/CVF CVPR, 2020, pp. 10781–10790.

[13] G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, 2008.

[14] "Forsyth–Edwards Notation," Wikipedia, The Free Encyclopedia.

[15] A. Buslaev, A. Parinov, A. Khvedchenya, A. I. Lavrentyev, and S. A. Kalinin, "Albumentations: Fast and Flexible Image Augmentations," Information, vol. 11, no. 2, 2020.

[16] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[17] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proc. ICLR, 2015.

[18] "Android Debug Bridge (ADB)," Android Developers, https://developer.android.com/studio/command-line/adb.