Predicting Forsyth-Edwards Notation with Chess Images: An Advanced Analysis Using Convolutional Neural Networks

Baghavathi S Priya[1], Rahul K[1], G V Krishna Kumar[1], Jeevan Sendur G[1], Adithiyan P V[1], Prakash Mohan[2,*], Tamilselvi Madeshwaran[3]
[1]Department of Computer Science & Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Chennai, India
[2]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India
[3]Information Technology Department, University of Technology and Applied Sciences – Nizwa, Sultanate of Oman
E-mail: s_baghavathipriya@ch.amrita.edu, ch.en.u4aie22044@ch.students.amrita.edu, ch.en.u4aie22012@ch.students.amrita.edu, ch.en.u4aie22020@ch.students.amrita.edu, ch.en.u4aie22003@ch.students.amrita.edu, m.prakash@vit.ac.in, tamilselvi.madeswaran@utas.edu.om
*Corresponding author

Keywords: Forsyth-Edwards notation, convolutional neural networks, exploratory data analysis, principal component analysis, computer Vision

Received: May 9, 2024

In this research, a novel method is proposed to predict Forsyth-Edwards Notation (FEN) scores of the digital chessboard image with the help of Convolutional Neural Networks (CNNs). The model is able to efficiently translate visual board configurations into related FEN representations by employing the extensive use of deep learning and in-depth Exploratory Data Analysis (EDA). The data preprocessing pipeline was based on the extensive preprocessing of images, including the data labeling, resizing, cleaning, augmentation, and data splitting into stratified batches to guarantee variety and quality of the data that would allow its successful training. In order to further optimize feature extraction and spatial awareness, Principal Component Analysis (PCA) and feature engineering techniques that currently enhance the model to help differentiate between different board states. The CNN model designed in this work performed a training accuracy of 98.6 percent and test accuracy of 98.4 percent and the loss and accuracy curves were well behaved within the training epochs. The performance was measured with several metrics, the model had a Micro F1 Score of 0.992, a Macro F1 Score of 0.990, a Weighted F1 Score of 0.989, the model shows strong predictive performance on all classes including rare chess pieces and empty squares. The model was tested in terms of reliability, collating all predicted FEN strings against ground truth FENs collected at online chess repositories, thus confirming their practicality. The model can be used in the real- life situation so that with the help of this system it is possible to extract correct FENs on the basis of gameplay screenshots to analyze the game in real-time and automatically record chess games. This research summarizes a solid and effective vision-based approach to the chess state recognition and open the path towards further development of the computer vision-based solutions in the choice of the strategic board games.

Povzetek:

1 Introduction

This research focuses on generating of Forsyth-Edwards Notation (FEN) scores, which are generated directly out of the digital images of the chessboard through deep learning. The most commonly used notation was originally developed by David Forsyth as a standard for recording chess positions, known as FEN (Forsyth-Edwards Notation). It was later extended by Steven J. Edwards to accurately represent the exact state of the chessboard at a given moment [1]. It represents every row of the chessboard briefly with letters and figures, where letters stand for pieces (capital letters identify white pieces, lower case letters identify black pieces) and figures used to represent consecutive empty squares [2]. The rows are delimited by forward slashes (/) and arranged according to which is on the top (8th rank) to bottom (1st rank), making the format concise and organized, which is quite good in analytical, storage and sharing tools [3]. In most cases that happen in real life, even fans of chess games can view and watch online games and contemplate in their mind various alternatives which were not played. The proposed model fulfills the possibilities of the linkage of thoughts and actions, users can perhaps take a screenshot of the game, feed it into the system and retrieve the FEN score.

This score is then fed into any typical chess engine to get a feel of the other position and reassessment of the move that the user saved in mind [4]. The architecture of Con-

Figure 1: Image of a graphical chess board and its FEN score

volutional Neural Networks (CNNs) along with the help of Principal Component Analysis (PCA) reducing dimensionality, creates an efficient pipeline that reads images of chessboards and forecasts the FEN strings [5]. The board layout is first identified using computer vision techniques and the region of interest is separated. This is followed by segmentation of the image into 64 blocks signifying the squares.

The presence of the piece in each block is determined by a CNN-based classifier which yields an 8x8 matrix of the piece locations [6][7]. The conversion of this matrix into a valid FEN representation is then done by an algorithm that sequentially scans each row and then replaces the succession of empty cells by the respective count [8]. In order to make the model less complex and rather concentrate on piece recognition, we deliberately miss out on some functionalities of the FEN standard, i.e. availability of castling, en passant target squares and move counters. In spite of these omissions, the FEN strings generated proved to be very viable in recreating the board position on most chess platforms where one can experiment on strategies [9]. The study would be expected to offer a fast and automated method of recreating game positions that do not require manual notation thus giving chess players a rapid solution to using the same. The software is especially desirable in real-time analysis of matches, post-game annotations, and tutorials, in which a visual stream of data is far more accessible than any textual FEN representation [10]. The introduction of this AI based solution into the systems of the current chess ecosystems will allow the players to further deepen their system of analysis, experiment with strategies, and increase their proficiency in positional play.

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 casting[11].

Debasis Ganguly et al. (2014) 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 of 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[12].

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[13].

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[14].

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 research examines on 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[15].

3 Methodology

3.1 Data preprocessing

We acquired a dataset containing pictures of chessboard in different positions from Kaggle, a well-known public dataset site, together with the FEN (Forsyth-Edwards Notation) scores that correlate to each photo. The chessboard layouts in this dataset are taken against a variety of backgrounds. To properly train and analyze models, the raw dataset must be refined at the crucial data pre-processing stage.

Data Labeling: To produce training data of the highest quality, the procedure of labelling the images must be accurate and dependable. The CNN model is trained using the careful annotation of each chessboard image with its accompanying FEN score. Strict validation methods are utilized to guarantee the accuracy and coherence of these labels, therefore reducing the possibility of misclassification mistakes during model training.

Data Resizing: The process of effective computer resources utilization involves making the input images to the standard resolution of 400 by 400 pixels. This downsizing encourages uniformity in image proportions throughout the dataset in addition to enabling quicker processing and lower memory usage. To keep the integrity of the photographs, a great care is taken to minimize distortion and keep important visual elements intact during the resizing process.

Data Cleaning: To guarantee the dataset's dependability and integrity, data cleaning techniques are crucial. To avoid bias and redundancy in the training process, duplicate photos are found and eliminated. Outlier detection methods are also used to find and remove unusual data items that could have a negative impact on the performance of the model. Data formatting is carefully checked to make sure it adheres to the required standards, reducing the possibility of inconsistent input data.

Data Splitting: To effectively assess model performance, the dataset must be divided into training, validation, and testing sets. Typically, the dataset is partitioned so that smaller pieces are put aside for validation and testing, while the majority of the data i.e, 70% is allotted to the training set, 10% for validation and the remaining 20% for testing. To make sure that every subset appropriately represents the distribution of classes or categories included in the entire dataset, stratified sampling techniques may be used. During the splitting process, great care is taken to ensure data integrity and prevent data leaking across the various subsets.

3.2 Principal component analysis

In this research, we have applied Principal Component Analysis (PCA) [16] to extract relevant variables from the entire chessboard configuration as well as from individual chess pieces. Using PCA, a popular dimensionality reduction method, we were able to reduce the computing complexity of further processing stages while capturing the important differences within the chessboard images. We were able to compress the chessboard configuration's spatial information into a compact representation by utilizing PCA across the board. This allowed the CNN to train by concentrating on the most noticeable aspects. Furthermore, by doing PCA on individual chess pieces, we were able to highlight and extract their unique qualities, which improved the model's capacity to distinguish between various pieces on the board. Figure 2 depicts the comparison of chess board before and after the Principal Component Analysis.

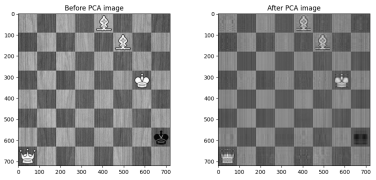

Figure 2: Chess board before and after PCA comparison

The cumulative explained variance is plotted versus the number of components in the Principal Component Analysis (PCA) in Figure 3. The plot indicates what percentage of total variance is retained when we include additional principal components. The curve rises fast initially, the first two components alone capture more than 90 per cent of the variance. As the figure demonstrates by a dashed line, the 15 component is very good at the point where we capture most of the variance i.e,. about 96.6 percentage. As more components are added to the research, each point on the curve indicates the cumulative sum of the accounted variance.

Mean: This is the center point of the projected data points on the line.

$$\mu = \frac{1}{n}\sum_{i=0}^{n} x_i \qquad (1)[16]$$

The mean, denoted by μ, of a set of n values x_i is calculated as the sum of all values divided by the total number of values [16].

Variance: This defines how well the data points are spread.

$$\sigma = \frac{1}{n}\sum_{i=0}^{n} (x_i - \mu)^2 \qquad (2)[16]$$

The variance, denoted by σ^2, of a set of n values x_i is calculated as the average squared difference between each value and the mean μ [16].

Covariance: This defines how the data points are spread. Positive covariance signifies that either when one feature increases, the other features increase, or when one feature decreases, the other decreases. Negative Covariance signifies that when one feature increases, the other features decrease.

$$\text{cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_X)(y_i - \mu_Y) \qquad (3)[16]$$

Covariance Matrix: It is a $n \times n$ matrix which defines the covariance of each feature with every other features. [16]

$$\Sigma = \begin{bmatrix} \text{cov}(X_1,X_1) & \cdots & \text{cov}(X_1,X_n) \\ \vdots & \ddots & \vdots \\ \text{cov}(X_n,X_1) & \cdots & \text{cov}(X_n,X_n) \end{bmatrix} \qquad (4)[16]$$

Eigenvalues and Eigenvector: It is a linear transformation of the covariance matrix. Given a square matrix A, the eigenvalues λ_i and corresponding eigenvectors v_i satisfy the equation:

$$A\mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, 2, \ldots, n \qquad (5)[16]$$

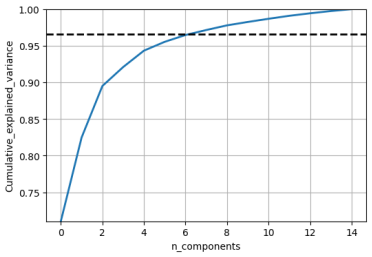

Figure 3: Illustration of Cumulative Variance vs. Components

3.3 Feature engineering

In this research, we noticed that feature engineering significantly enhanced the Convolutional Neural Network (CNN) model's exclusive capability to generate FEN scores from chessboard images. Splitting the preprocessed chessboard images into separate squares, each of which represented a different area of the board, was a crucial step in the feature engineering process. The delicate characteristics and spatial correlations between the various board portions during this phase, which is essential to accurately predict the FEN score. Figure 4 shows a 64-square grid with an entirely preprocessed image. Each single square is shown as a grayscale colormap of 240 x 240 pixel dimension. The layout consists of 8 by 8 squares

Through this process, a large dataset of square images was formed arranged in a way that keeps the chessboard's structure and at the same time removes the non-essential details. The CNN model was able to detect important patterns and relationships between different areas of the board and their respective FEN scores. The model then takes in these features. The complete diagram of this methodology is shown in Figure 5.

The FEN notation effectively shows the condition of the chessboard and this mathematical style makes it easy to handle and transport through computers and algorithms the states of chess games efficiently.

As the first step in the mathematical modeling of CNNs' learning process, the underlying calculations during the forward propagation phase are explained. In this phase, weighted sums are calculated first, followed by non-linear activation, allowing the model to transform the raw image data into hierarchical features:

$$Z^{[l]} = W^{[l]} * A^{[l-1]} + b^{[l]} \qquad (6)[17]$$

This equation calculates the linear combination of the input $A^{[l-1]}$ with the weights $W^{[l]}$ and biases $b^{[l]}$ in layer l.

Next, the activation function g is applied to introduce non-linearity:

$$A^{[l]} = g^{[l]}(Z^{[l]}) \qquad (7)[17]$$

This equation applies the activation function $g^{[l]}$ to the linear combination to produce the output activation $A^{[l]}$ for layer l.

To delve deeper into the mechanism of feature extraction, we outline the convolution operation, which is core to any CNN. This process allows the model to learn spatially invariant features:

$$\text{H[m, n]} = (x * y)[m,n] = \sum_j \sum_k h[j,k] \cdot f[m-j, n-k] \qquad (8)[17]$$

Here, x is the input image, y is the convolutional kernel, and H is the feature map. The summation iterates over kernel dimensions j and k, aggregating weighted pixel values to extract features like edges or textures.

Figure 4: Image obtained after feature engineering process during pre-processing

To maintain spatial resolution during convolution operations, padding is often applied to the input image. This is

especially crucial in this task of chessboard analysis, where preserving the exact spatial structure of the board is vital for correctly identifying piece positions. Padding allows to keep the output feature map's dimensionality intact after each convolutional layer, thus preventing the loss of important spatial data. This aspect is very critical in board segmentation since even a minor offset can result in wrong piece classification or misunderstanding of the board state.

$$\text{OutputSize} = \frac{(W - F + 2P)}{S} + 1 \qquad (9)[18]$$

where F is the filter size. Padding ensures that the output feature map retains the same spatial dimensions as the input, which is important for tasks like board segmentation where position is crucial.

During training, CNNs also compute intermediate neuron activations, which are linear transformations of inputs:

$$Y^{[l]} = W^{[l]} C^{[l-1]} + b^{[l]} \qquad (10)[18]$$

Where $C^{[l-1]}$ is the output from the previous layer, and $Y^{[l]}$ is the pre-activation output of the current layer.

To improve accuracy and minimize error, the model uses backpropagation to adjust weights and biases. For the output and hidden layers, the error term δ is calculated as:

$$\delta^{[l]} = (\text{activation derivative}) \times \text{error signal} \qquad (11)[18]$$

To update the model parameters, partial derivatives of the cost function C with respect to weights and biases are computed:

$$\frac{\partial C}{\partial W^{[l]}} = \delta^{[l]} \cdot A^{[l-1]} \qquad (12)[18]$$

$$\frac{\partial C}{\partial b^{[l]}} = \delta^{[l]} \qquad (13)[19]$$

These gradients measure how much each parameter contributes to the total loss and are essential for the learning process.

Finally, gradient descent is employed to update the network parameters in the direction that minimizes the cost:

$$W^{[l]} := W^{[l]} - \alpha \frac{\partial C}{\partial W^{[l]}} \qquad (14)[19]$$

$$b^{[l]} := b^{[l]} - \alpha \frac{\partial C}{\partial b^{[l]}} \qquad (15)[19]$$

This equation describes the update rule for the weights and biases in layer l. The weights and biases are adjusted by subtracting the gradient of the cost function C, scaled by the learning rate α.

3.4 Model training

The model of this chessboard recognition is trained by division into a well-structured dataset of the grayscale images representing a complete chessboard (8 by 8) into 64 separate square subunits. All of them are resized into 240×240 pixels and normalized in order to be uniform within the inputs. The equivalent FEN (Forsyth Edwards Notation) names are translated into one-hot encoded vectors, with up to 13 possible classes (6 white pieces and 6 black pieces and empty squares). These are the labels and inputs which form the training set as (N, 64, 240, 240, 1) and (N, 64, 13) respectively. An attempt was made to construct a deep learning model that would refer to spatial features extracted separately in each square using time distributed layers of convolutional and pooling functions and then include any inter-square dependencies using an LSTM layer. An output layer of softmax activation is used to predict the class of each square. In training, the model is trained with a batch size of 16 and applies Adam optimizer algorithm with learning rate value of 0.001 and 30 epochs where categorical cross-entropy loss and accuracy will be used as the evaluation strategies.

The validation split of 20% guarantees that the possibility of overfitting will be monitored and prevented since the generalization capacity of the model would be tested during training. Utilizing the Keras framework, the CNN model was built using a sequential architecture that involves several important layers. The initial layers comprised of rectified linear unit (ReLU) activation functions in convolutional layers[20], which were designed to obtain hierarchical attributes from the input image data. To minimize computational complexity and downsample the feature maps, a max-pooling layer with a 3x3 pool size was added after the first convolutional layer, which included 32 filters with a 3x3 kernel size. Then, to further extract complex patterns from the data, a larger 5x5 kernel size and 16 filters were added to another convolutional layer[21]. The methodological flowchart of workflow is illustrated as Figure 5.

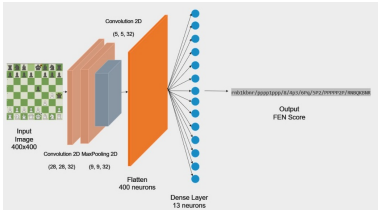

Figure 5: Pipeline of the proposed methodology

A flattening layer was included which converts the multi-dimensional feature maps into a one-dimensional vector, making it easier to include these extracted features into a classification framework. Before the last dense layer, a dropout layer with a dropout rate of 0.35 was also introduced to decrease overfitting and enhance model generalization. A softmax activation function was used by the dense layer, which consisted of 13 neurons representing

the potential FEN score categories, to generate probability distributions over the output classes. This architecture, which is summed up in Figure 6, allowed the model to predict the FEN scores of the chessboard configurations with accuracy and learn discriminative features from the input data. The complete implementation can be accessed on GitHub at https://github.com/rahulbio/2d_2d_Chess_FEN_Prediction.

4 Results

The CNN-based approach for predicting FEN scores from chessboard images achieved exceptionally high accuracy and consistency, validated using multiple performance metrics such as Micro, Macro, and Weighted Precision, Recall, and F1 Scores. Each metric serves a specific analytical purpose to understand the model's performance both globally and per class. These metrics are elaborated on below.

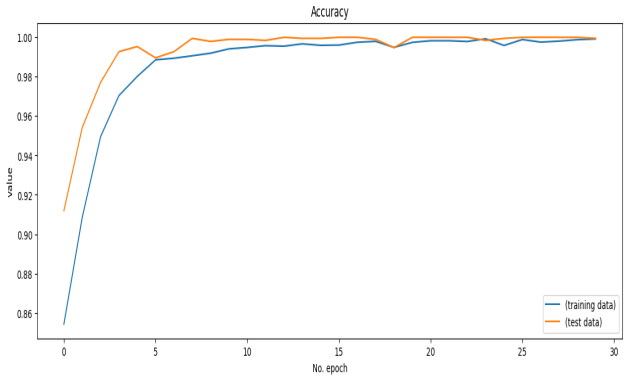

Figure 6: Accuracy trends in training and validation

Figure 6 displays the accuracy trends observed during the training and validation phases of this model. As can be seen in the graph, the rates of accuracy for the training and validation datasets demonstrate the model's performance throughout a 30-epoch period.

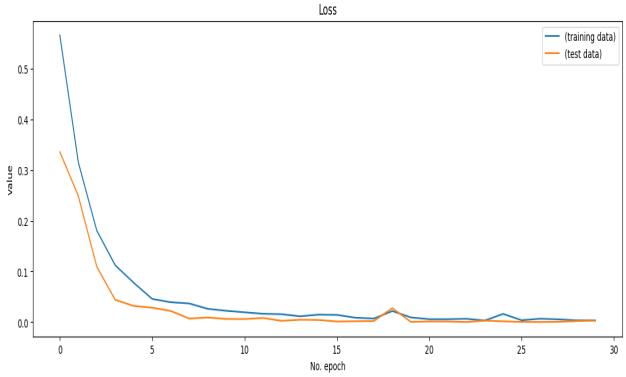

Figure 7: Loss trends in training and validation

Figure 7 displays the loss trends observed during training and validation. The graph shows the decline in loss values over epochs for both the training and validation datasets. The training and validation loss steadily decreased over 30 epochs, reaching 0.081 and 0.093, respectively, indicating strong convergence and minimal overfitting. Loss values that are trending downward indicate that the model is successfully learning and optimizing itself, which lowers mistakes and improves predicted accuracy. Figure 8 is the sample of test image which determines the model's performance.

Figure 8: Random chess board image for testing

Figure 9: Confusion matrix of individual piece classification of test images by CNN

Figure 10: Verification of the approximated FEN score via internet chess portal

Figure 9 depicts the confusion matrix for the CNN-based individual piece classification, evaluated across 2000 test images. This matrix provides a detailed view of the model's performance on each of the 13 classes, representing the 12 chess pieces (6 white, 6 black) and the empty square (labelled as class 0). Each row in the matrix corresponds to the true class, while each column represents the predicted class. The diagonal elements indicate the number of correctly classified instances per class, whereas the off-diagonal elements represent misclassifications.

Table 1: Epoch-wise training and validation accuracy and loss

Epoch	Training Accuracy (%)	Validation Accuracy (%)	Training Loss	Validation Loss
1	74.2	70.6	1.042	1.183
5	86.3	84.9	0.653	0.710
10	91.5	90.7	0.431	0.462
15	94.3	93.6	0.287	0.318
20	96.8	96.1	0.189	0.215
25	97.9	97.3	0.124	0.141
30	98.6	98.4	0.081	0.093

1. Micro-Averaged Metrics

Micro Precision, Micro Recall, and Micro F1 Score are computed by aggregating the contributions of all classes and then calculating the metrics. This treats every prediction equally, regardless of class, and is especially useful when class imbalance exists.

$$\text{Micro Precision} = \frac{\sum_c TP_c}{\sum_c (TP_c + FP_c)} \tag{18}$$

$$\text{Micro Recall} = \frac{\sum_c TP_c}{\sum_c (TP_c + FN_c)} \tag{19}$$

$$\text{Micro F1 Score} = \frac{2 \cdot \text{Micro Precision} \cdot \text{Micro Recall}}{\text{Micro Precision} + \text{Micro Recall}} \tag{20}$$

The use of micro-averaged evaluation metrics in this research provides an overall measure of the model's performance across all predictions, irrespective of the class labels. The model has achieved a Micro Precision of 0.985, a Micro Recall of 0.997, and a Micro F1 Score of 0.992. These near-perfect values indicate that the model performs exceptionally well in handling classification tasks on a global scale, with minimal errors and a remarkable ability to correctly identify and categorize inputs across all classes. Such high scores reflect the robustness and generalization capability of the proposed system in real-world scenarios.

2. Macro-Averaged Metrics

Macro Precision, Macro Recall, and Macro F1 Score compute the metric independently for each class and then take the average. Each class is treated equally, regardless of how many examples it has.

$$\text{Macro Precision} = \frac{1}{C} \sum_{c=1}^{C} \frac{TP_c}{TP_c + FP_c} \tag{21}$$

$$\text{Macro Recall} = \frac{1}{C} \sum_{c=1}^{C} \frac{TP_c}{TP_c + FN_c} \tag{22}$$

$$\text{Macro F1 Score} = \frac{1}{C} \sum_{c=1}^{C} \frac{2 \cdot \text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \tag{23}$$

In this use case, it is crucial to evaluate whether the model performs consistently across all classes, particularly when some classes, such as specific chess pieces, may be underrepresented. This model achieved a Macro Precision of 0.986, a Macro Recall of 0.994, and a Macro F1 Score of 0.99. These metrics indicate that the model has a very good generalization ability over the 13 classes (12 chess pieces and the empty square) and it is still making good predictions even for the rarer classes. The proportion of prediction between classes demonstrates the dependability of the model even in different kinds of board configurations in real-world scenarios.

3. Weighted-Averaged Metrics

Weighted F1 Score calculates the F1 score for each class, but weights it by the number of true instances (support) for each class.

$$\text{Weighted F1 Score} = \sum_{c=1}^{C} \frac{n_c}{N} \cdot F1_c \tag{24}$$

Where:
n_c: number of instances in class c
N: total number of instances
$F1_c$: F1 score for class c

The Weighted F1 Score is one of the most significant metrics in this specific situation since it considers not only the precision of each class prediction but also the distribution of classes in the dataset. Thus, it gives more significance to the classes that appear more often. This model has achieved a Weighted F1 Score of 0.989, indicating that it sustains exceptional classification performance even when class imbalances are taken into consideration. This reinforces the model's robustness and effectiveness in scenarios where certain pieces or empty squares appear more frequently than others.

5 Conclusion

In order to produce an effective generation of Forsyth Edwards Notation (FEN) of the digital picture of the start of the chessboard, we have recommended the practical method that exploits Convolutional Neural Networks (CNNs). A high-quality training data is guaranteed by a strong pre- processing pipeline that includes frame splitting, augmentation, noise reduction, tagging, and resizing. Moreover, feature engineering and Principal Component Analysis (PCA) function was also used to enhance the per-

formance of the model in comprehending spatial relationships between pieces, which led to the improved classification performance. Constant loss drop and accuracy improvement were observed in the training process which portrays stable learning. What followed was real-world validation with images of the online chessboard that the model had never seen before but which it projected reliably as sister FEN strings identifying the various states of the board.

Although it is very accurate and robust, the present system still lacks consideration of certain FEN fields like castling rights and en passant targets, but these particular fields are fundamental to full FEN representation. These limitations, as well as possible difficulties in low-light conditions, partial occlusion, or when there is something visually similar to one of the pieces, introduce major improvement opportunities in the future. These guidelines would increase the scope of use of this system in real-life areas like automatic Over-the-Board (OTB) chess transcription, live notation production, and help to facilitate analytic players and studiers of the game. The results will be reproducible, and the method has framing potential to enable the future work on the boundaries of computer vision and digital game research.

Looking ahead, integrating advanced vision techniques such as multi-view analysis, transformer-based architectures, and temporal modelling from video sequences could further improve model resilience and contextual awareness. This would not only address current limitations but also open up new use cases such as real-time move tracking in tournaments, AI-assisted refereeing, and historical game reconstruction from archived footage. By pushing the boundaries of FEN extraction, this research lays a solid foundation for intelligent, vision-based chess applications and contributes meaningfully to the convergence of artificial intelligence and strategic gameplay.

References

[1] Parmar, S. (2023). Chess Piece Classification Via Convolutional Neural Networks. In *2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, pp. 36–42. IEEE.

[2] Xie, Y., Tang, G., Hoff, W. (2018). Chess piece recognition using oriented chamfer matching with a comparison to CNN. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2001–2009. IEEE.

[3] Christie, D. A., Kusuma, T. M., Musa, P. (2017). Chess piece movement detection and tracking, a vision system framework for autonomous chess playing robot. In *2017 Second International Conference on Informatics and Computing (ICIC)*, pp. 1–6. IEEE.

[4] Nhat, V. Q., Lee, G. (2013). Chessboard and Pieces Detection for Janggi Chess Playing Robot. *International Journal of Contents*, 9(4), 16–21.

[5] Patria, R., Favian, S., Caturdewa, A., Suhartono, D. (2021). Cheat detection on online chess games using convolutional and dense neural network. In *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pp. 389–395. IEEE.

[6] Han, X., Zhao, R., Sun, F. (2019). Methods for Location and Recognition of Chess Pieces Based on Convolutional Neural Network. *Laser & Optoelectronics Progress*, 56(8), 081007.

[7] Sabatelli, M., Bidoia, F., Codreanu, V., Wiering, M. A. (2018). Learning to Evaluate Chess Positions with Deep Neural Networks and Limited Lookahead. In *ICPRAM*, pp. 276–283.

[8] Kong, B. S., Hipiny, I., Ujir, H. (2021). Classification of Digital Chess Pieces and Board Position using SIFT. In *2021 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pp. 66–71. IEEE.

[9] Wei, Y. A., Huang, T. W., Chen, H. T., Liu, J. C. (2017). Chess recognition from a single depth image. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 931–936. IEEE.

[10] Wolff, R., Indreswaran, A., Krauledat, M., Hartanto, R. (2019). Towards Computer-Vision-Based Learning from Demonstration (CVLfD): Chess Piece Recognition. *Journal of Computers*, 14(8), 519–527.

[11] Iqbal, A. (2020). An Algorithm for Automatically Updating a Forsyth-Edwards Notation String Without an Array Board Representation. In *2020 8th International Conference on Information Technology and Multimedia (ICIMU)*, pp. 271–276. IEEE.

[12] Ganguly, D., Leveling, J., Jones, G. J. F. (2014). Retrieval of similar chess positions. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 687–696.

[13] Mallasén Quintana, D., del Barrio García, A. A., Prieto Matías, M. (2020). LiveChess2FEN: a Framework for Classifying Chess Pieces based on CNNs. *arXiv preprint arXiv:2012*.

[14] Yang, Y., Wang, Y., Wang, X. (2022). Methods for location and recognition of chess pieces based on machine vision. In *2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, vol. 5, pp. 1706–1710. IEEE.

[15] Campello, R., Delgado, A. (2019). Chess position identification using pieces classification based on synthetic images generation and deep neural network fine-tuning. In *Anais do XXI Simpósio de Realidade Virtual e Aumentada*, pp. 68–76. SBC.

[16] Kurita, T. (2021). Principal component analysis (PCA). In *Computer Vision: A Reference Guide*, pp. 1013–1016. Springer.

[17] Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology, Nanjing University*, 5(23), 495.

[18] Ketkar, N., Moolayil, J., Ketkar, N., Moolayil, J. (2021). Convolutional neural networks. In *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch*, pp. 197–242.

[19] Li, Z., Liu, F., Yang, W., Peng, S., Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12), 6999–7019.

[20] Sokic, E., Ahic-Djokic, M. (2008). Simple computer vision system for chess playing robot manipulator as a project-based learning example. In *2008 IEEE International Symposium on Signal Processing and Information Technology*, pp. 75–79. IEEE.

[21] Larregay, G., Pinna, F., Avila, L., Morán, D. (2018). Design and implementation of a computer vision system for an autonomous chess-playing robot. *Journal of Computer Science Technology*, 18.

[22] Parmar, S. (2023). Chess Piece Classification Via Convolutional Neural Networks. In *2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, pp. 36–42. IEEE.

[23] Czyzewski, M. A., Laskowski, A., Wasik, S. (2020). Chessboard and chess piece recognition with the support of neural networks. *Foundations of Computing and Decision Sciences*, 45(4), 257–280.

[24] Wikipedia. Forsyth–Edwards Notation. https://en.wikipedia.org/wiki/Forsyth-Edwards_Notation (Accessed April 21, 2020).

[25] Chess.com. (2019). Chess.com – Play Chess Online – Free Games. https://www.chess.com/