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

1st Dr.S.Baghavathi Priya
Associate Professor
Dept. of Computer Science & Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Chennai, India
s_baghavathipriya@ch.amrita.edu

2nd Rahul K
Dept. of Computer Science & Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Chennai, India
ch.en.u4aie22044@ch.students.amrita.edu

3rd G V Krishna Kumar
Dept. of Computer Science & Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Chennai, India
ch.en.u4aie22012@ch.students.amrita.edu

4th Jeevan Sendur G
Dept. of Computer Science & Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Chennai, India
ch.en.u4aie22020@ch.students.amrita.edu

5th P V Adithiyan
Dept. of Computer Science & Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham
Chennai, India
ch.en.u4aie22003@ch.students.amrita.edu

Abstract

With the use of convolutional neural networks (CNNs), our study offers a novel method for predicting Forsyth-Edwards Notation (FEN) scores from digital chessboard images. With the use of sophisticated deep learning techniques and exploratory data analysis (EDA), our model is able to convert chessboard configurations into FEN scores with precision. By carefully labeling, resizing, cleaning, augmenting, and splitting data, among other preprocessing steps, we guarantee the diversity and high quality of our dataset for reliable model training. Furthermore, to improve the discriminative capacity of the model, Principal Component Analysis (PCA) and feature engineering are used to improve feature extraction and spatial correlations within chessboard images.With convolutional and loss layers, our CNN architecture performs exceptionally well, achieving 98.6% training accuracy and 98.4% testing accuracy. The model's learning process and consistent development over epochs are demonstrated by loss and accuracy trends, which validate the model's dependability and effectiveness. The accuracy and applicability of the model for real-world scenarios are further validated by comparing predicted FEN scores to ground truth values.The proposed approach shows promise for a number of useful applications, such as real-time analysis of online matches and automated chess game record keeping. Our methodology helps users make better strategic analysis and decision-making in chess by allowing them to compute FEN scores from screenshots of active games. Our study brings up new possibilities for investigation and growth in this field and advances computer vision-based chess analysis.

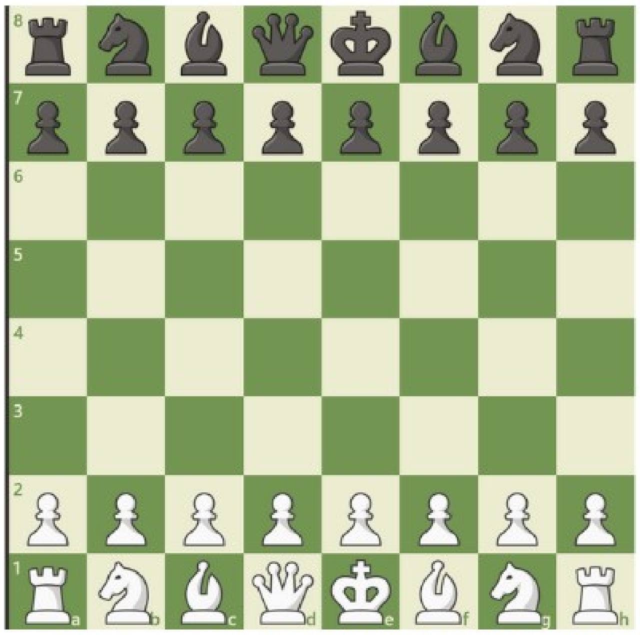
Index Terms

Forsyth-Edwards Notation,Convolutional Neural Networks,Exploratory Data Analysis,Principal Component Analysis,Computer Vision

I. Introduction

FEN scores are useful for analyzing chess games. The proposed model uses Exploratory Data Analysis and Convolution Neural Networks. It takes the digital chess board as input and it gives the FEN score as output. Suppose, a person watches an online chess match and during the mid-game, the person thinks of one move but the chess player makes another move, the person would want to see how the game would proceed if the move the person thought was played. Hence, he can take a screenshot, and then he could pass it into the deep learning algorithm and generate the FEN score. The person can then use this generated FEN score on any chess website and replicate the chess board and the person can further analyze the move the person wants to make.

Forsyth-Edwards Notation(FEN)[1] score, developed by journalists David Forsyth and Steven J. Edwards around 19th century, is a representation used to describe the current position of the chess game. FEN portrays a brief way of denoting chess board arrangement with the help of alphabets and numbers. As a typical chess board consists of eight rows, the FEN score is scripted in such a way that each rows are separated by forward slashes(/). The rows in this notation are arranged from top to bottom. The first row from the top of a chess board is the first row in the FEN notation. Similarly, the successive rows are represented by



rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR

Fig. 1. Image of a Graphical chess board and its FEN score

FEN notation, where a ”/” distinguishes one row from the another. The FEN notation makes use of six alphabets which are K, Q, R, N and B respectively. Here, K, Q, R, N and B represent King, Queen, Rook, Knight and Bishop pieces correspondingly. The capital letters denote white colored pieces and the lowercase letters denote the black colored pieces. The numbers from 1-8 are used to indicate the blank spaces in the chess board in the respective row. While calculating the FEN score, each cell in the row should be checked for pieces and should be filled with appropriate piece notation as mentioned earlier. If a cell is empty, the FEN notation applies an equivalent number to indicate the empty cells present in the row. For example, if there are three successive blank cells in a row, The FEN notation is represented by ”3” followed by other pieces present in that specific row. In the proposed approach, the castling and en passant properties of the FEN notation are neglected, which enables a clear illustration of chess board and pieces. A straightforward incorporation of the suggested approach into the existing chess platforms helps players strategize their plan and experiment their ideas with their moves from that particular position in the game.

FEN scores are useful for analyzing chess games. The proposed model uses Exploratory Data Analysis and Convolution Neural Networks[2][3]. It takes the digital chess board as input and it gives the FEN score as output[4][5]. Suppose, a person watches an online chess match and during the mid-game, the person thinks of one move but the chess player makes another move, the person would want to see how the game would proceed if the move the person thought was played. Hence, the player can take a screenshot, and then he could pass it into the deep learning algorithm[6] where computer vision[7] is used to identify the position and types of pieces[8][9] and generate the FEN score. The person can then use this generated FEN score on any chess website and replicate the chess board[10] and the person can further analyze the move the person wants to make. Fig.1 shows the example of a Chess board with its corresponding FEN notation.

II. 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 study 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].

III. Methodology

A. Data pre-processing

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, lighting situations, and camera angles. To properly train and analyze models, the raw dataset must be refined at the crucial data pre-processing stage.

1) Data Labeling: To produce training data of the highest quality, the procedure of labelling the images must be accurate and dependable. Our 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.

2) Data Resizing: Achieving efficient use of computer resources involves scaling up the input photos to a 400 by 400-pixel standard resolution. 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.

3) 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.

4) 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.

B. Principal Component Analysis

In this study, we have applied Principal Component Analysis (PCA)[18][19] 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.

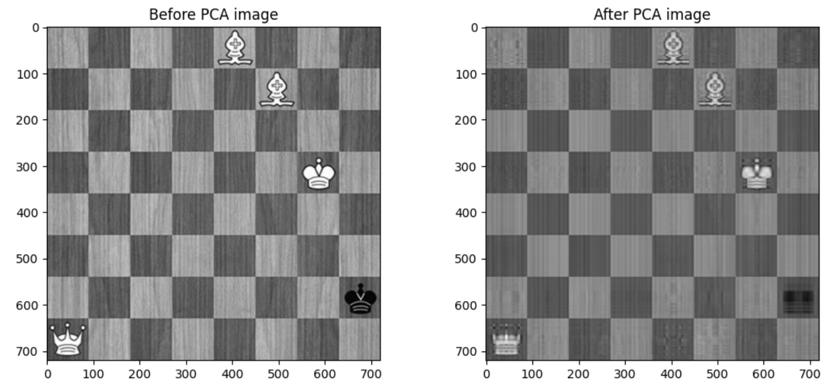


Fig. 2. Chess board before and after PCA comparison

The cumulative explained variance is plotted versus the number of components in the Principal Component Analysis (PCA) in Figure 3. Utilizing 15 Components Preserves 96.6% Features. As more components are added to the study, each point on the curve indicates the cumulative sum of the accounted variance. The proportion of variance kept by the chosen number of components is indicated by the horizontal dashed line, which represents the percentage of reserved features.

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

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

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.

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

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{2}$$

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.

3) 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.

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

The covariance between two variables X and Y, denoted by $\text{cov}(X, Y)$, of a set of n paired values (x_i, y_i) is calculated as the average of the product of the deviations of each variable from their respective means μ_X and μ_Y.

4) Covariance Matrix: It is a nxn matrix which defines the covariance of each feature with every other features.

$$\Sigma = \begin{bmatrix} \text{cov}(X_1, X_1) & \text{cov}(X_1, X_2) & \cdots & \text{cov}(X_1, X_n) \\ \text{cov}(X_2, X_1) & \text{cov}(X_2, X_2) & \cdots & \text{cov}(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(X_n, X_1) & \text{cov}(X_n, X_2) & \cdots & \text{cov}(X_n, X_n) \end{bmatrix} \tag{4}$$

The covariance matrix Σ represents the covariance between each pair of variables in a set of n variables $X_1, X_2, \ldots, X_n$.

5) Eigenvalues and Eigenvector: It is a linear transformation of the covariance matrix. Given a square matrix A, the eigenvalues λ_i and corresponding eigenvectors $\mathbf{v}_i$ satisfy the equation:

$$A\mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, 2, \ldots, n \tag{5}$$

This equation describes the relationship between a square matrix A, its eigenvalues λ_i, and corresponding eigenvectors $\mathbf{v}_i$. Each eigenvalue λ_i corresponds to an eigenvector $\mathbf{v}_i$, such that when the matrix A is multiplied by the eigenvector, the result is a scaled version of the same eigenvector.

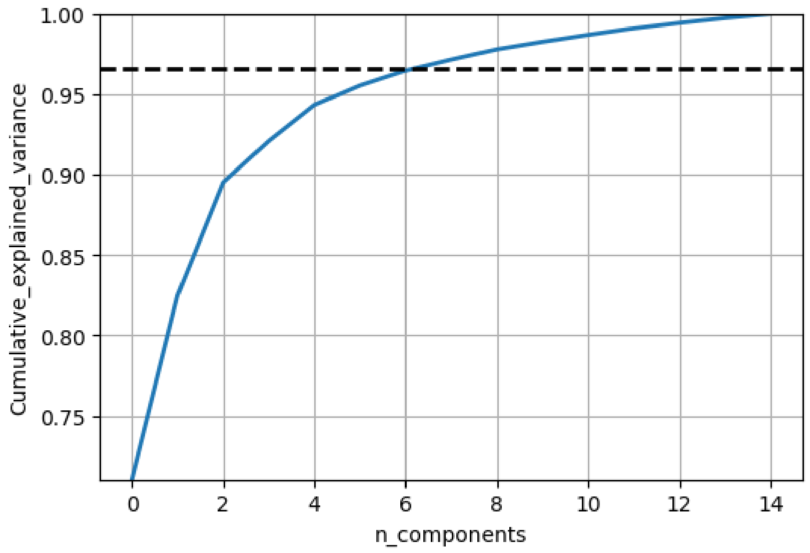


Fig. 3. Illustration of Cumulative Variance vs. Components

C. Feature Engineering

In this research, we noticed that feature engineering significantly enhanced our 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 our feature engineering process. We were able to record 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 square is displayed as a grayscale colormap with 240 x 240 pixel dimensions. The grid is set up in an 8 by 8 configuration. We created an extensive dataset by carefully assembling the square images into an array, which preserves the chessboard's structural features while eliminating unnecessary information. Our CNN model was capable of identifying significant patterns and connections among various board regions and their corresponding FEN scores. These features are then passed into the CNN model. The complete diagram of our methodology is shown in Figure 5.

The FEN notation efficiently represents the state of the chessboard and this mathematical form enables chess game states to be processed and conveyed in computer systems and algorithms in an effective way. Some important formulae used in CNNs are given below

1) Forward propagation: The forward propagation equations for a Convolutional Neural Network (CNN) are as follows:

$$Z^{[l]} = W^{[l]} * A^{[l-1]} + b^{[l]} \tag{6}$$

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

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

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

2) Kernel Convolution:

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

x represents the input image, and y represents our kernel. The result matrix's row and column indexes are denoted by m and n, respectively.

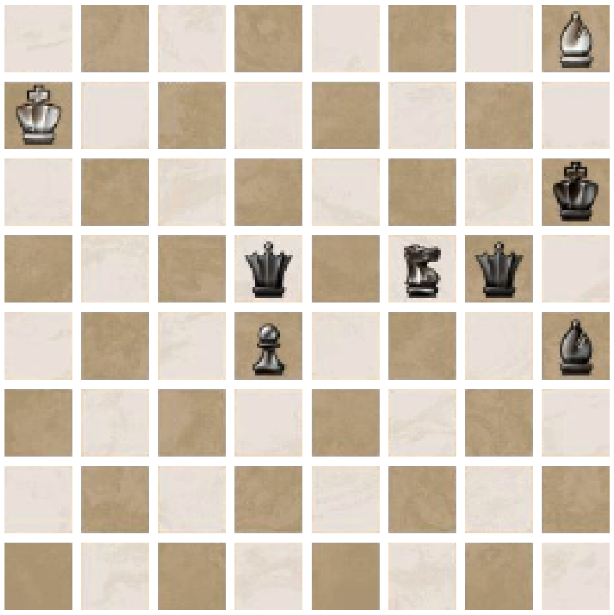


Fig. 4. Image obtained after feature engineering process during Pre-Processing

3) Padding Formula:

$$p = \frac{g-1}{2} \tag{9}$$

where p denotes padding and g denotes the filter dimension , which is usually odd

4) Intermediate Neuron Value: The intermediate neuron value $Z^{[l]}$ can be calculated using the following equation:

$$Y^{[l]} = W^{[l]}C^{[l-1]} + b^{[l]} \tag{10}$$

where, Y is the intermediate value, W is the weight of the connected neurons, C is the value of the neuron and b is the bias of the layer.

5) Backpropagation formulae for CNNs:

$$\delta_j^l = \begin{cases} (\hat{y}_j - y_j) \cdot \sigma'(z_j^L), & \text{if } l = L \text{ (output layer)} \\ (\sum_k \delta_k^{l+1} \cdot w_{kj}^{l+1}) \cdot \sigma'(z_j^l), & \text{if } l \text{ is a hidden layer} \end{cases} \tag{11}$$

$$\frac{\partial C}{\partial w_{ij}^l} = a_i^{l-1} \cdot \delta_j^l \tag{12}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{13}$$

The jth neuron's error in layer l is represented by δ_j^l. The expected output of the jth neuron is $\hat{y}_j$. The real output is y_j. The jth neuron in its output layer receives its input from z_j^L. In the lth layer, the jth neuron is connected to the $(l+1)$th layer, where $w_{kj}^{l+1} \cdot \sum_k \delta_k^{l+1}$ denotes the weighted total of errors from that layer. The activation of the ith neuron in the $(l-1)$th layer is denoted by a_i^{l-1}. We denote the cost function as C. $w_{ij}^l = \frac{\partial C}{\partial}$ is the partial derivative of the cost function C with respect to the weight w_{ij}^l that links the jth and ith neurons in the lth and $(l-1)$th layers, respectively. The partial derivative of the cost function C with respect to the bias term b_j^l for the jth neuron in the lth layer is denoted by the expression $\frac{\partial C}{\partial b_j^l}$.

6) Gradient Descent: The gradient descent update rules for a Convolutional Neural Network (CNN) are as follows:

$$W^{[l]} = W^{[l]} - \alpha \frac{\partial J}{\partial W^{[l]}} \tag{14}$$

This equation describes the update rule for the weights $W^{[l]}$ in layer l. The weights are adjusted by subtracting the gradient of the cost function J with respect to the weights, scaled by the learning rate α.

$$b^{[l]} = b^{[l]} - \alpha \frac{\partial J}{\partial b^{[l]}} \tag{15}$$

This equation describes the update rule for the biases $b^{[l]}$ in layer l. The biases are adjusted by subtracting the gradient of the cost function J with respect to the biases, scaled by the learning rate α.

D. Model training

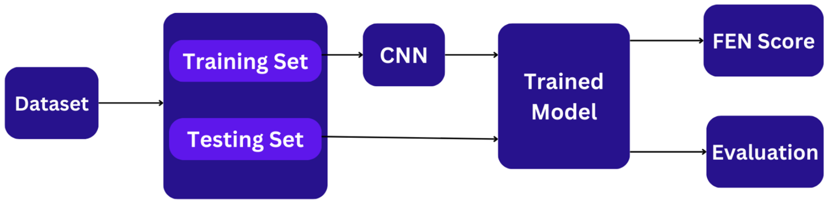


Fig. 5. Flowchart of our proposed methodology

In this study, we utilized a convolution neural network architecture to build a prediction model that could extract FEN scores from chessboard images. 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. The Architecture flowchart is illustrated as figure 6. A

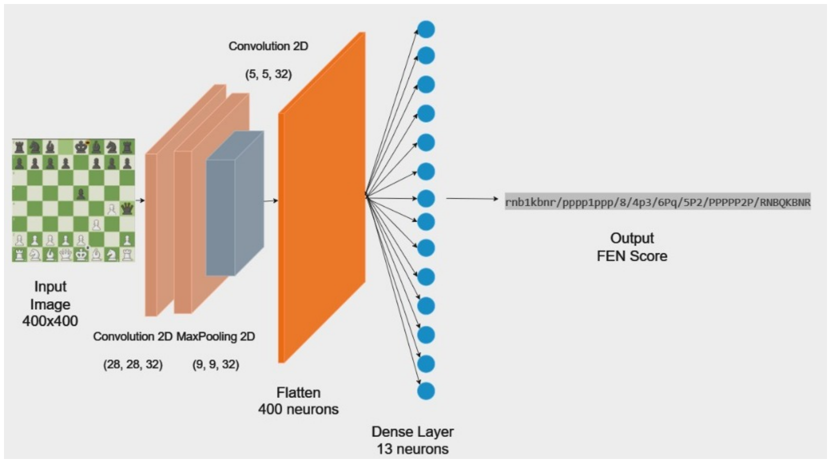


Fig. 6. Architecture of the CNN model

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.

IV. Results

In the end, our CNN-based technique for predicting FEN scores from chessboard images produced remarkable results. Our model performs remarkably well, with high testing and training accuracy rates of 98.4% and 98.6%, respectively, along with robust losses and accuracy trends. By comparing predicted FEN scores to ground truth values, we may further establish our model's accuracy and dependability. These findings demonstrate the effectiveness of our methodology and pave the way for future advances in computer vision-based chess analysis approaches.

Figure 7 displays the accuracy trends observed during the training and validation phases of our 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.

$$\text{Accuracy} = \frac{(TP + TN)}{\text{Total Predictions}} \tag{16}$$

where,

$$\text{Total Predictions} = (TN + FN + TP + FP)$$

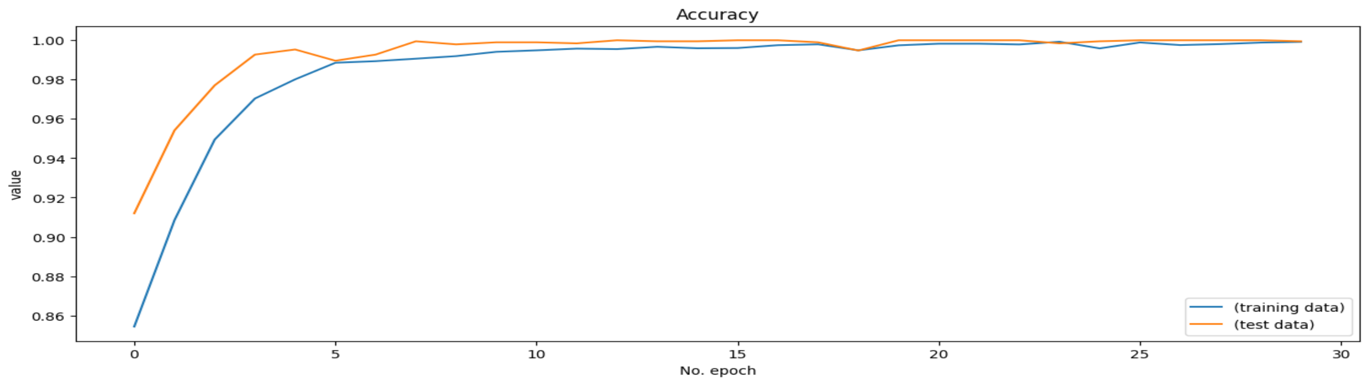


Fig. 7. Accuracy Trends in Training and Validation

Figure 8 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. Loss values that are trending decrease indicate that the model is successfully learning and optimizing itself, which lowers mistakes and improves predicted accuracy[22].

$$\text{Loss} = -\sum_{x}[x \cdot \log(\hat{x}) + (1 - x) \cdot \log(1 - \hat{x})] \tag{17}$$

The true label (ground-truth) of an occurrence is denoted by x, and the predicted probability (output) for that instance is represented by $\hat{x}$.

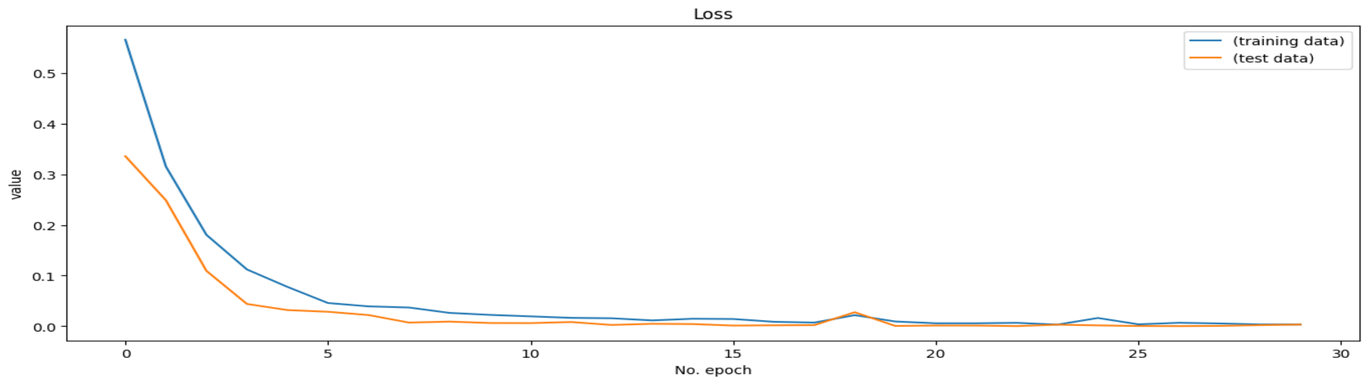


Fig. 8. Loss Trends in Training and Validation

We used an image of a randomly generated checkerboard (Figure 9) to assess our model's prediction ability. The picture depicts a situation in which people take photos of chess games that are in progress in order to analyze and forecast the outcome[23].

Figure 10 shows how our model's estimated FEN scores are verified by comparing them with FEN values from internet chess sources. The precision and dependability of our model's predictions are confirmed by the alignment of predicted FEN scores with the actual ones confirmed online. This validation highlights how our method may be used practically in real-world chess analysis situations, where the ability to anticipate FEN scores accurately is crucial for making strategic decisions[24].

Lastly, we have demonstrated impressive results using our CNN-based approach to FEN score prediction from chessboard images. Our algorithm works very effectively with good testing and training accuracy rates of 98.4% and 98.6%, as well as strong loss and accuracy trends. The accuracy is verified by testing it in a real life situation and the output is verified by pasting the generated FEN score in a chess analysis website, in this case, we used chess.com to verify the FEN score.

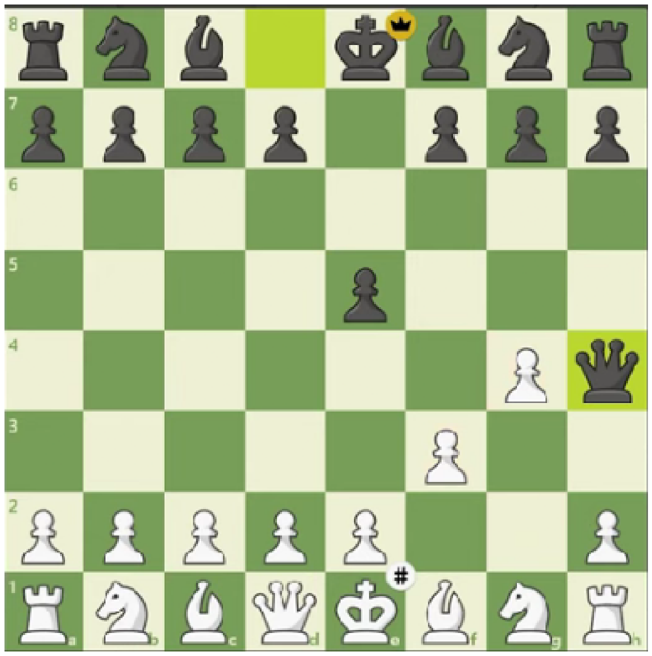


Fig. 9. Random Chess Board Image for Testing

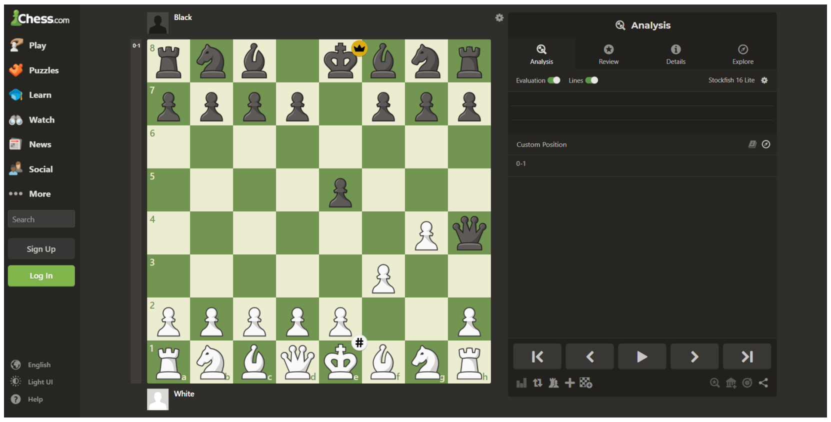


Fig. 10. Verification of the approximated FEN score via internet chess portal

V. Conclusion

We propose an efficient method which uses convolutional neural networks, commonly known as CNNs, to predict Forsyth Edwards Notation (FEN) notation from the images of chessboard. We performed advanced techniques of pre-processing which involved splitting, augmentation, cleaning, tagging, and resizing, to ensure that it was of utmost quality. The usage of Principal Component Analysis (PCA) and feature engineering, which helps to obtain related data from chess board layouts and individual pieces, improved the ability of CNN for classification and the capacity to identify spatial relations. Model's learning process was revealed through loss and accuracy trends which were gradually increase throughout epochs, ensuring its efficiency and reliability.

The model's performance was validated by comparing the predicted FEN scores with real values. Additionally, an online image of a chessboard was selected randomly and was used to explain how efficiently the model handled input data comprised of different formats and accurately identified the FEN values. In addition to major advances in computer vision-based chess analysis, our method is found to have potential applications in real-world scenarios, some of which includes automating live chess game recording and enabling online real-time play using physical chessboards. The consistency of our method indicates that it has the power to enhance computer vision and chess analysis techniques, creating new opportunities for deeper understanding of this marvelous game

References

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

[2] Xie, Youye, Gongguo Tang, and William Hoff. ”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, 2018.

[3] Christie, Dennis Aprilla, Tubagus Maulana Kusuma, and Purnawarman Musa. ”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, 2017.

[4] Nhat, Vo Quang, and GueeSang Lee. ”Chessboard and Pieces Detection for Janggi Chess Playing Robot.” International Journal of Contents 9, no. 4 (2013): 16-21.

[5] Patria, Reyhan, Sean Favian, Anggoro Caturdewa, and Derwin Suhartono. ”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, 2021.

[6] Han, Xie, Rong Zhao, and Fusheng Sun. ”Methods for Location and Recognition of Chess Pieces Based on Convolutional Neural Network.” Laser & Optoelectronics Progress 56, no. 8 (2019): 081007.

[7] Sabatelli, Matthia, Francesco Bidoia, Valeriu Codreanu, and Marco A. Wiering. ”Learning to Evaluate Chess Positions with Deep Neural Networks and Limited Lookahead.” In ICPRAM, pp. 276-283. 2018.

[8] Kong, Brandon Sean, Irwandi Hipiny, and Hamimah Ujir. ”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, 2021.

[9] Wei, Yu-An, Tzu-Wei Huang, Hwann-Tzong Chen, and JenChi Liu. ”Chess recognition from a single depth image.” In 2017 IEEE International Conference on Multimedia and Expo (ICME), pp. 931-936. IEEE, 2017.

[10] Wolff, Regina, Anoshan Indreswaran, Matthias Krauledat, and Ronny Hartanto. ”Towards Computer-Vision-Based Learning from Demonstration (CVLfD): Chess Piece Recognition.” J. Comput. 14, no. 8 (2019): 519-527.

[11] Iqbal, Azlan. ”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, 2020.

[12] Ganguly, Debasis, Johannes Leveling, and Gareth JF Jones. ”Retrieval of similar chess positions.” In Proceedings of the 37th international ACM SIGIR conference on Research development in information retrieval, pp. 687-696. 2014.

[13] Mallasén Quintana, David, Alberto Antonio del Barrio García, and Manuel Prieto Matías. ”LiveChess2FEN: a Framework for Classifying Chess Pieces based on CNNs.” arXiv e-prints (2020): arXiv-2012.

[14] Yang, Yongfeng, Yaoqi Wang, and Xiaopeng Wang. ”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, 2022.

[15] Campello, Rafael, and Afonso Delgado. ”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, 2019.

[16] Kagkas, Dimitrios, Despina Karamichailidou, and Alex Alexandridis. ”Chess Position Evaluation Using Radial Basis Function Neural Networks.” Complexity 2023 (2023).

[17] Baxter, Jonathan, Andrew Tridgell, and Lex Weaver. ”Learning to play chess using temporal differences.” Machine learning 40 (2000): 243-263.

[18] Wold, Svante, Kim Esbensen, and Paul Geladi. ”Principal component analysis.” Chemometrics and intelligent laboratory systems 2, no. 1-3 (1987): 37-52.

[19] Chatfield, Chris. ”Exploratory data analysis.” European journal of operational research 23, no. 1 (1986): 5-13.

[20] Sokic, Emir, and Melita Ahic-Djokic. ”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, 2008.

[21] Larregay, Guillermo, Federico Pinna, Luis Avila, and Daniel Morán. ”Design and implementation of a computer vision system for an autonomous chess-playing robot.” Journal of Computer Science Technology 18 (2018).

[22] Wölflein, Georg, and Ognjen Arandjelović. ”Determining chess game state from an image.” Journal of Imaging 7, no. 6 (2021): 94.

[23] Czyzewski, Maciej A., Artur Laskowski, and Szymon Wasik. ”Chessboard and chess piece recognition with the support of neural networks.” Foundations of Computing and Decision Sciences 45, no. 4 (2020): 257-280.

[24] Wikipedia, ”Forsyth-Edwards Notation,” available at https://en.wikipedia.org/wiki/Forsyth-Edwards$_Notation, accessed on 21 April 2020.$

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