FEN Chess Position Classification Using CNN

Snir Lugassy	Gal Oshri	Elon Amram
206312506	311363873	205679780

Abstract

Forsyth-Edwards Notation (FEN) is a common notation for chess board state representation. In FEN each piece is represented using a unique letter, The white pieces are denoted using upper-case letters and the black pieces using lower-case letters.

We will train a Convolutional Neural Network to classify the unique state of a Chess board in FEN notation given images taken from online Chess website.

Forsyth-Edwards Notation Definition

The full FEN will describe many details about a certain game, we will consider only notation used to describe the pieces placement in the board.

The pieces placement is defined from the white player perspective, using 6 groups of characters separated by a delimiter. We will use '-' as a delimiter (adopted from the dataset) although the original definition suggests the usage of '/' as a delimiter.

Each group of characters corresponds to a row in the board from top to bottom, each character corresponds to a piece or space in the row from left to right according to the following dictionary:

"p" → Pawn
"n" → Knight
"b" → Bishop
"r" → Rook
"q" → Queen
"k" → King
1-8 → Number of empty squares

White pieces are represented using upper-case letters and the black pieces using lower-case letters.

The state of the board in Figure 1 using FEN is:

$$1B1B2K1-1B6-5N2-6k1-8-8-8-4nq2$$

Chess Position Dataset

Our dataset consists of **100,000 images** of chess boards taken from the online website “Chess.com”, with a size of 400×400 pixels. The boards have variety of colors, and the pieces has a variety of both shape and color.

We used 80,000 images for training and 20,000 images for testing.

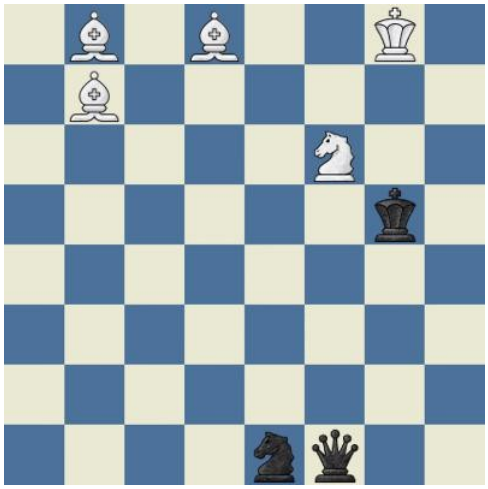

Figure 1: Example board image

The state of each board in FEN is stored in the file name of the image file.

Data Processing and Transformations

Since the Chess Positions Dataset was already cropped and labeled properly, the amount of processing over the data was minimal.

Image transformations

Each image file was converted into a vector with the following shape: $8\times8\times3\times50\times50$

The first and second dimensions represents a position on the chess board, the third dimension is the color channel of the image (RGB). The fourth and fifth dimensions are the color values of the image.

The color values were normalized into the range $[0,1]$ using $f(x)=\frac{x}{255}$.

Label transformations

The image labels, holding the FEN notation for each image, were converted into a label matrix holding values from the label set $L=\{0,1,2,\ldots,12\}$. Each label corresponds to a possible piece on the chess board including an empty cell.

In order to convert FEN into matrix and matrix into FEN, we wrote the helper functions “fen2matrix” and “matrix2fen”. The 8x8 label matrix holds 0 if the cell is empty, otherwise it holds the piece number (predefined index from letters to number and from numbers to letters).

Model

In our model the FEN classification task is split into smaller tasks, in which we classify the label of each cell using a CNN. The CNN's input is 50x50 images of board cells, and the output a weight vector over the label set (vector in R^{13}).

In order to predict the label of each 50x50 image we fed into the CNN we used the *log-softmax* function, which outputs a probability distribution over the label set, then the maximal entry index in the vector was considered as the predicted label.

The CNN Consists of 2 convolutional layers and 2 fully connected hidden layers. The convolutional layers contains 12 filters with size 4x4. The first fully connected layer size is 50 and the second layer size is 13 (the output layer).

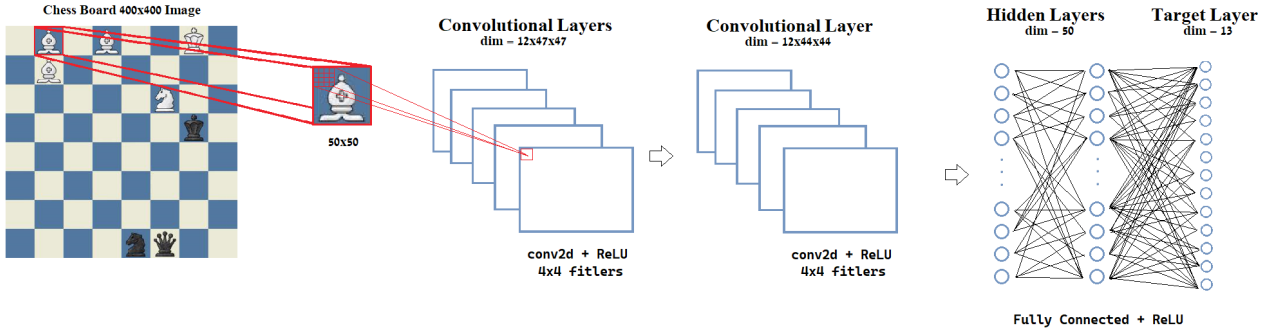

Figure 2: Network Architecture

Inference

Inferring the FEN notation state of a given board image is achieved in 2 steps:

1. Predict the piece in each cell of the board using the trained CNN
2. Convert the prediction matrix to FEN notation string (using *matrix2fen*)

The above can be described using the following pseudo-code:

```
board2fen(board):
      define mat[8][8];
      for each row i in board:
            for each cell j in row:
                  mat[i][j] ← argmax log(softmax(CNN(board[i][j])))
      return matrix2fen(mat)
```

Results
The model accuracy over the 20,000 test images is **99.93%**, based on the number of correctly predicted board cells divided by the total number of cells (number of boards times 64).

We were interested in another evaluation criteria, therefore we chose to compare the ground-truth label and the predicted label in FEN notation using Hamming Distance.

Hamming Distance: The Hamming distance of 2 strings with the same length is the number of positions in which the strings holds different symbols.

In order to consider strings with different length we padded them into the same length.
The following is the Hamming distance results between the predicted FEN label and the ground-truth label over 20,000 test images:

Hamming Distance	Number of samples	Percent of samples
0	19,249	96.24%
1	630	3.15%
2	93	0.465%
3	19	0.095%
3+	9	0.045%

Discussion
We were interested in building a model involving games and Neural Networks, Therefore, we are pleased with our FEN classification model and the results.

Given more time, we would implement the following improvements:

1. Generalization for different images taken from multiple websites: In order to improve the generalization we would use the following techniques:
- **Random Crop**: randomly crop the image into a smaller image
- **Random Rotation**: randomly rotate the image
- **Channel Random Dropout**: randomly zero-out entire channel (with low probability)

2. Classifying real chess board pictures: Combined with another network, our model can be extended to classify pictures of real chess boards.

Performance: The model performed beyond our initial expectations, we assumed that different colors and shapes of pieces will lead to lower accuracy than the actual test accuracy.

Applications: Our model can be used for further applications that involves chess games, e.g., prediction of the next move. Moreover, Our model can be used for translating a current state of chess board into voice in order to improve the accessibility of online chess games.