Chess Positions - FEN Prediction with CNN

INTRODUCTION

This is a vision AI application through CNN (Convolutional Neural Network) that understands FEN positions of the given chess board by looking at it. One of the methods to solve solving this problem is to use a CNN that is fed by the chess board blocks divided into 64 squares represents alternative positions or pieces. In order to obtain the findings, we first encode the labels into classes, and then decoding the outcomes to recreate the images with expected FEN codes.

A common notation for defining a certain chess board position is the Forsyth-Edwards Notation (FEN). The goal of FEN is to offer all the information required to restart a game from a specific point. By the time of now, researchers who study machine learning have shown that it is possible to correctly guess a chess player's identity based on the sequence of positions from their chess games. The method utilized in the study has the potential to be widely used for identifying people based on a variety of activities.

CNN is a deep learning method that can take in an input picture, assign importance to various characteristics and objects in the image, and distinguish between them. In comparison to other classification methods, a convolutional network requires substantially less preprocessing. Convolutional networks can also learn these filters and attributes with enough training, unlike basic approaches where filters must be manually engineered.

FEN Samples

FEN begins by encoding the chessboard's information in rows from top to bottom. The individuals explaining the components are:

K:	White King	k:	Black King
Q:	White Queen	q:	Black Queen
R:	White Rook	r:	Black Rook
B:	White Bishop	b:	Black Bishop
N:	White Knight	n:	Black Knight
P:	White Pawn	p:	Black Pawn
1-8:	1 to 8 empty boxes		

The board on the left side above with purple and white squares' FEN code defines that;

From top rows to bottom, and left squares to right,

3 empty squares, black bishop, 4 empty squares,

2 empty squares, black knight, 5 empty squares,

1 empty square, white bishop, 6 empty squares,

7 empty squares, black bishop,

8 empty squares,

2 empty squares, black knight, 3 empty squares, white king, 1 empty square,

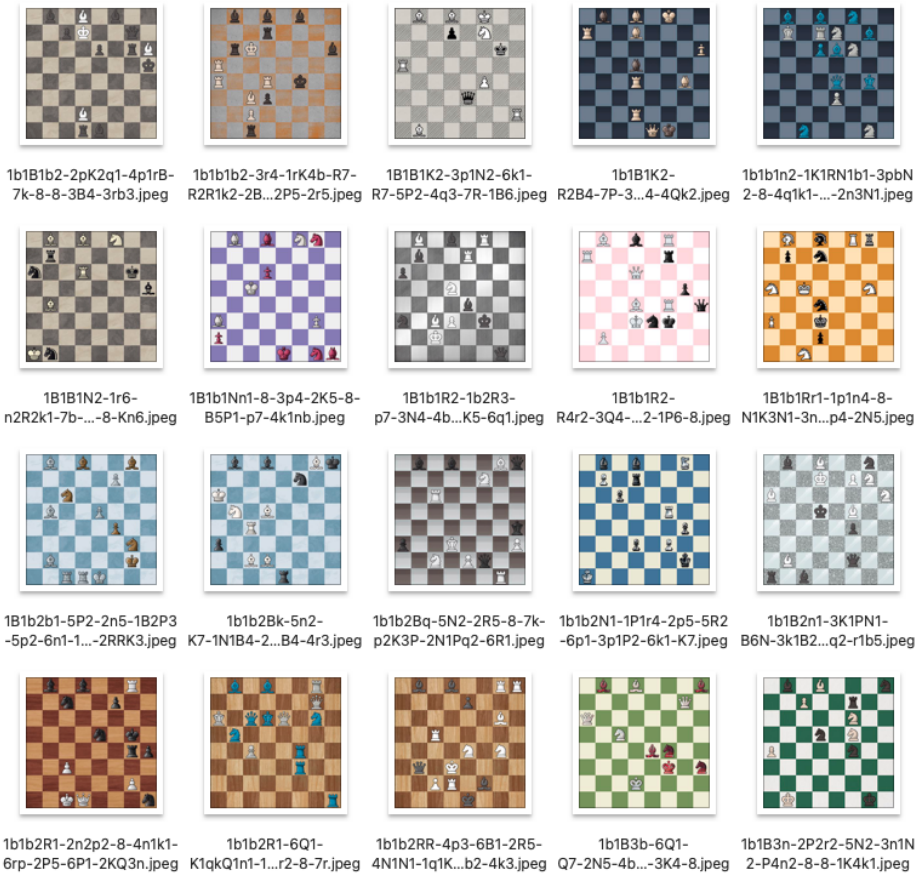
2 empty squares, black bishop, 5 empty squares,

6 empty squares, black king, white rook.

METHODOLOGY

Dataset Information

There are 100K chess board images of randomly produced 5 to 15 piece chess configurations, with two kings and 3–13 pawns/pieces each 896 different board/piece combinations are utilizing 32 different chess piece types and 28 different chess board designs. There are 400 × 400 pixels in each image. 80000 pictures comprise the training set. A test batch of 20,000 images.



The probability distribution used to create the pieces is as follows:

- 30% for Pawn

- 20% for Bishop
- 20% for Knight
- 20% for Rook
- 10% for Queen

2 Kings are guaranteed to be on the board.

The link to the dataset can be found here.

Model and Training Approach

After getting the details of the dataset and the model, the encoder and decoder functions were needed to be able to inspect the FEN codes. The data was mostly clean and has the same Ratios, Weights, and Sizes; thus, I should concentrate on the picture quality, the number of pixels per inch, and the method of equally dividing the blocks into 64 pieces while maintaining the form and the specific details of the individual pieces.

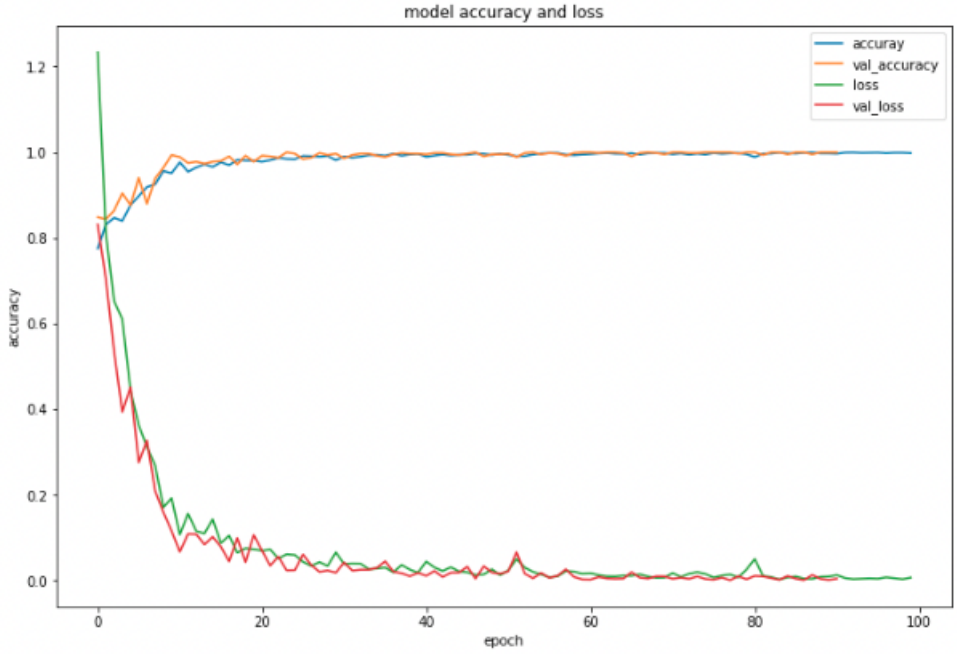
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 24, 24, 15)        195

 max_pooling2d (MaxPooling2D  (None, 12, 12, 15)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 5)         305

 flatten (Flatten)           (None, 605)               0

 dropout (Dropout)           (None, 605)               0

 dense (Dense)               (None, 13)                7878

=================================================================
Total params: 8,378
Trainable params: 8,378
Non-trainable params: 0
_________________________________________________________________
```

Two convolution layers with two distinct kernels are used in the model. One to concentrate on the individual components and the other to look more closely at the interactions between these two. The feature maps' size are reduced using MaxPooling, followed by a flattening layer and a SoftMax dense layer. During training, regularization was required, hence dropout is also utilized. The model is then built and trained, and its history and weights are kept for subsequent use. This way allowed to examine the accuracy and model loss evaluations for each epoch.

RESULTS

Thanks to the precise FEN encoder and decoder parts, the model was able to train quickly with high accuracy while having a loss close to zero after 40th epoch or so. Before all, less number of epochs are used to have a result, but the training was more performant and less time consuming after cutting it into more peaces. The accuracy has got above

0.90 levels in considerably early phase of the epoch evolution. The loss at the same time, went down from the levels of 1.2 to three decimals points in the end of the model training.



Predicted FENs

As it can be seem in here, the model gave an accurate result after applying methods as it shows in the plot above, result were going hand to hand when sample predictions checked.

CONCLUSION

The problem itself was involving technical parts as we have seen the theoretical aspects in the class. The applications of such a concept like this was pretty straightforward thanks to the libraries and the tools used. CNN itself was a model that can predict the problem in an efficient and fast way, it was surprising to see the evolution of the training even it was processing thousands of images at once. After all, it is good to know that there are use-cases like the involved problem, so it will be easier to apply a model like this in required occasions.

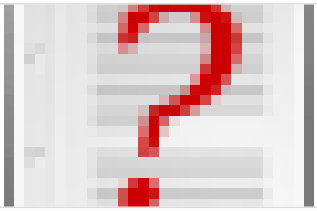
REFs

W https://en.wikipedia.org/wiki/Convolutional_neural_network

Forsyth-Edwards Notation - Wikipedia

Forsyth-Edwards Notation (FEN) is a standard notation for describing a particular board position of a chess game. The purpose of FEN is to provide all the necessary information to restart a game from a particular position. FEN is based on a system developed by Scottish

W https://en.wikipedia.org/wiki/Forsyth%E2%80%93Edwards_Notation



Chess Positions

Recognise chess position of 5-15 pieces

k https://www.kaggle.com/datasets/koryakinp/chess-positions