Chessboard recognition

Loris Giulivi

Introduction

Chess

Chess is a rather complex game, in order to improve, one must analyze board positions and to do so it is often useful to transfer the game state to a computer.

Board analysis

To transfer the board state, usually players must manually position the pieces in the virtual chessboard; this is tedious and time-consuming.

Project focus

With this work, we aim at speeding up the board analysis process by developing a tool able to automatically reconstruct the board state given a picture.

Related work

Previous work

Chessboard recognition is a computer vision problem that has not yet been efficiently solved. Previous studies either used some form of aid in the visual recognition task or had results that were not highly accurate.

Examples of other studies

Chessboard and chess piece recognition with the support of neural networks

Stockfish chess engine was used to improve chess piece prediction, together with statistical information from a large chess positions dataset, obtaining very good results albeit with higher computational costs.

ChessVision: Chess Board and Piece Recognition

In a very controlled environment, piece recognition was able to reach 85% accuracy.

Target workflow

Full-stack board recognition

Our project focuses on building a tool that can accurately reconstruct the chessboard state from a **single smartphone photo**.
This means that both the **position of the board** and the **positions of the pieces** on the board must be recognized.

Output

The output of the tool is a chess engine compatible **FEN string** that represents the board state.

Usability in field conditions

We assume the photo to be taken in relatively good conditions, but we allow for imperfect acquisition, so that the **smartphone may be hand-held and no additional equipment** is required.

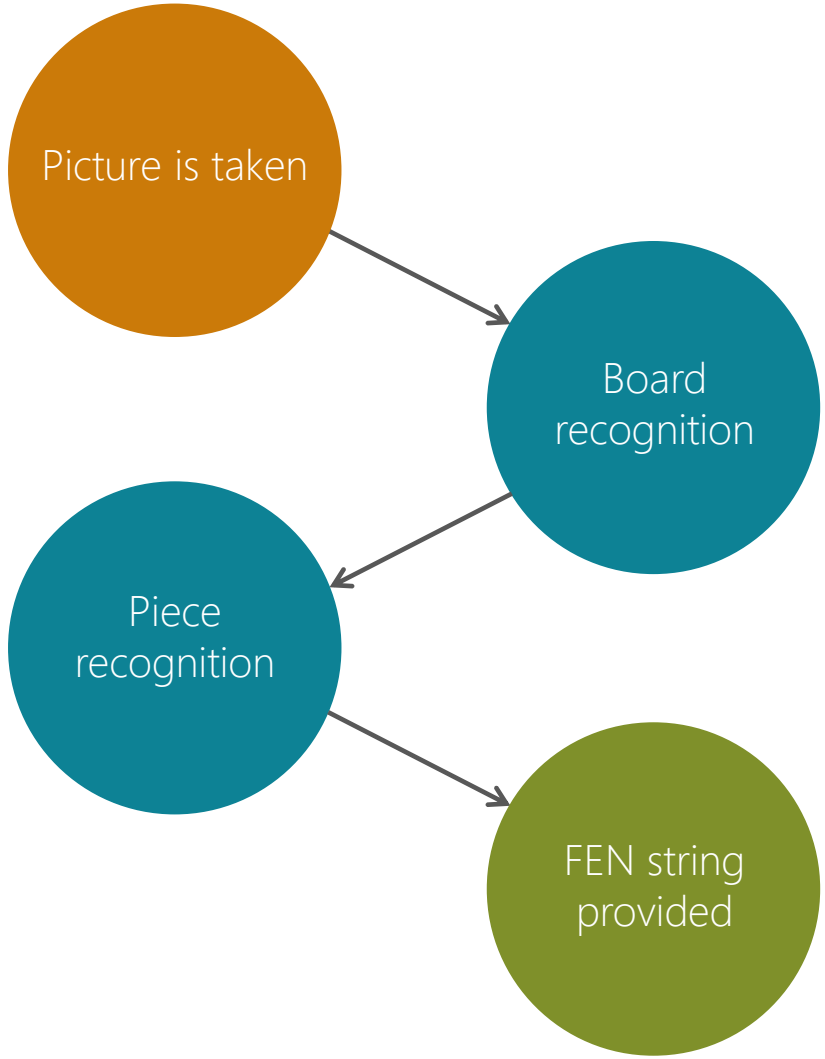
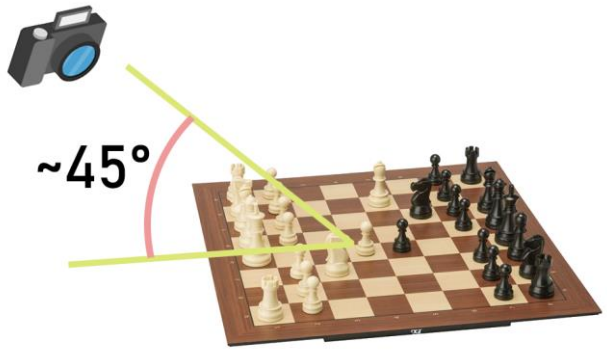


Photo acquisition

camera position

Pictures of the chessboard are taken so that the image plane is roughly parallel to one of the edges of the board, and with a pitch of roughly 45 degrees w.r.t. the board plane.



devices

To train and test the models used in this work, a Google Pixel 3A was used, but no calibration parameters were ever computed or used, so this work should work similarly with any standard smartphone camera.

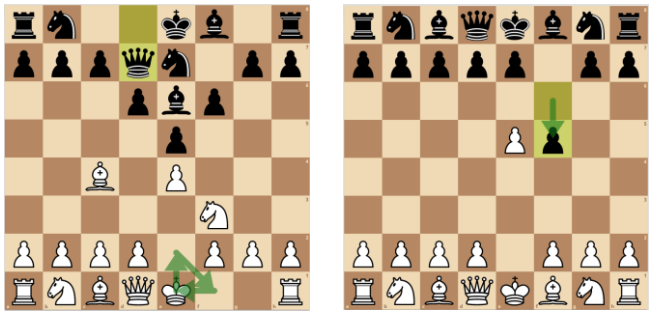
Lost data

Unobtainable information

Some characteristic of a chess game state cannot be entailed simply from a still image of the board. Therefore, some information is lost. This, however, has a negligible impact on the reconstruction process.

Castling righs, *en passant*

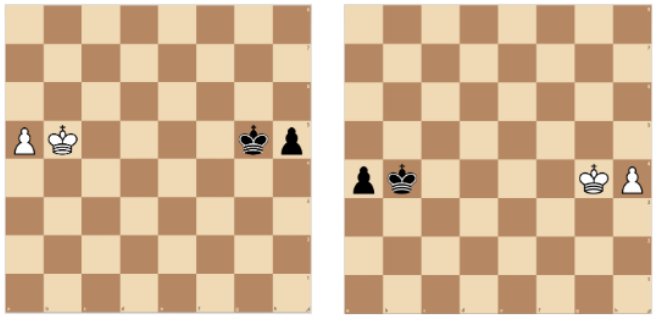
Castling and *en passant* are special moves in chess that are only available depending on the full move list of the game. Obviously, this information cannot be retrieved with a picture of the board.



Castling rights and en passant might be voided but we can't know from the board position alone.

Symmetries

Symmetries in the chessboard mean that some positions may be indistinguishable. In this cases, the program will have to choose which configuration to output without being sure of its correctness.



Symmetric positions are indistinguishable from each other

Threefold repetition

Finally, threefold repetition is a tie condition that depends on the whole move list and cannot be checked against the still board picture.

Approach

Geometry

Geometrical features alone are not enough to recognize pieces on the board, as pieces might be of the same material of the board and have the same colour and texture, making segmentation impossible.
Moreover, to allow this work to be generalizable to different sets of pieces that might have different geometrical properties, this approach was discarded for piece recognition.

Geometric properties, however, can and have been used to recognize the board location in the image.

Deep learning

A convolutional neural network was used to perform classification of the pieces of the board by using transfer learning on a resnet50 net.

Colour

Differences in colour shades have extensively been used to recognize the board location, the pieces side (white or black) and whether the picture was taken from the side or front of the chessboard. This process was always supported by clustering algorithms and is therefore independent of the specific colour of the chess set in use.

Dataset

Previous work

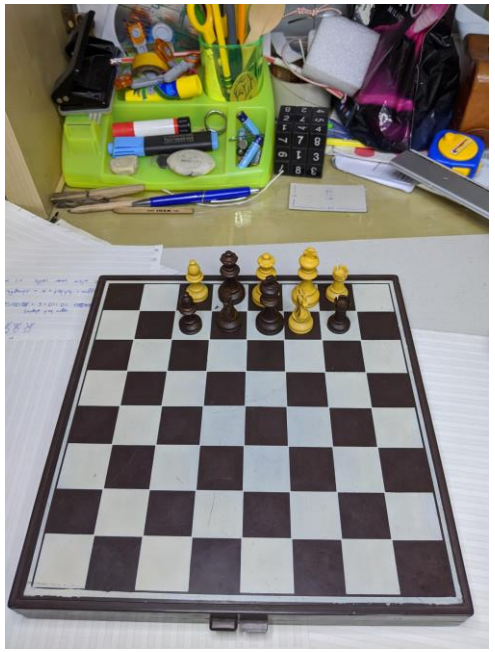
No public dataset was available that was large enough to train a CNN. To train the models used in this work, a **new dataset was created** from scratch.

Dataset contents

The dataset was populated with images taken with the same method explained before. The subject was a standard, low cost chessboard. The board was **randomly populated**, and pictures were taken in **different locations and with different lighting conditions**.

Pre-processing and labeling

The photos were pre-processed so that they could be cropped to fit single pieces in the chessboard. Afterwards, **each piece was labelled manually** using a purpose-built tool.



Sample image from the raw dataset

Dataset acquisition

Chessboard grid segmentation

For each picture in the dataset, the **four corners of the chessboard were manually selected** to accurately locate the board in the image.

After this process, the **image can be rectified** and supposing the board to be a square, we can subdivide the surface and obtain locations for all the squares in the 8x8 grid.

Manual selection of four corner points

Piece recognition

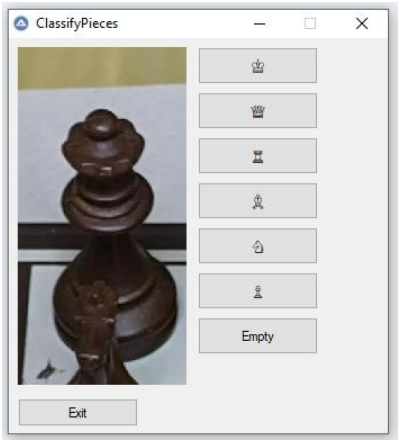
Using the information relative to the position of the board squares in the image, the **image was cropped for each piece** in it, creating snippets that could be fed to the network.

CNN Ready image crop
Note how the image base lines up with the square

Dataset acquisition

Labelling tool

To label the pieces, a tool was created that offers a GUI. This allows the user to quickly scan through all the generated snippets and assign them to the relative class, thus generating labelled samples to be fed to the network.



GUI of the labeling tool

Diamond crop

Experiment have shown that the network can be deceived in cases where the board is cluttered, and more than one piece appear in the snippet. To solve this issue, samples were further cropped into a diamond shape, to help the network "concentrate" on the square relative to the sample.

Diamond cropped sample

Dataset acquisition

Acquisition workflow

We summarize here the acquisition process for clarity

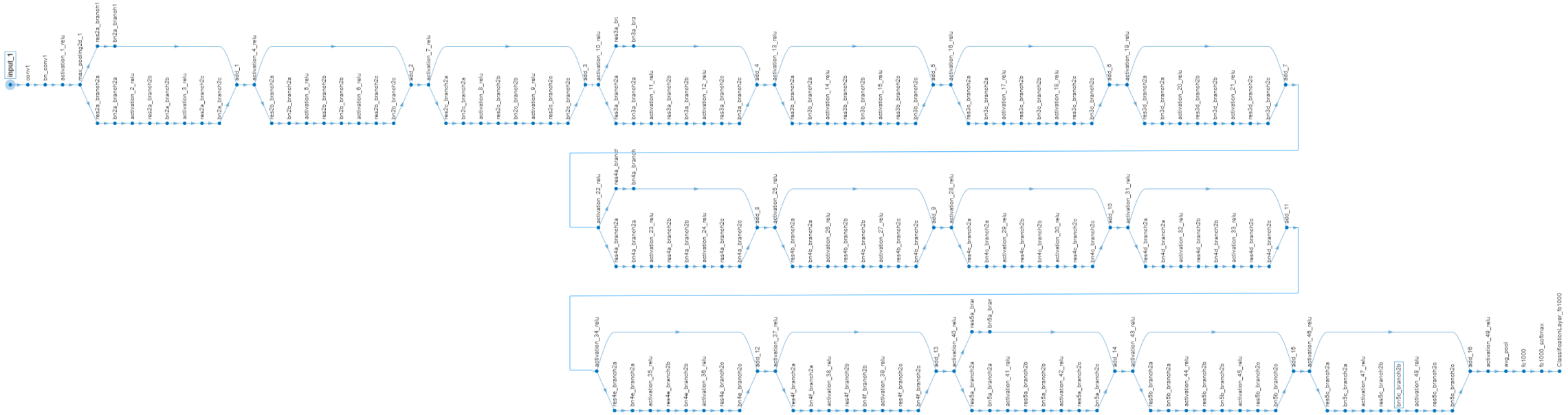
Classifier network

Model and training

A pre-trained ResNET 50 was taken and the last layer was replaced to perform transfer learning.

Samples were rescaled to 224x224 to fit to the network input and augmentation was performed in the form of random translations and rotations of the input image.

Training was performed using SGDM optimizer and cross entropy loss.

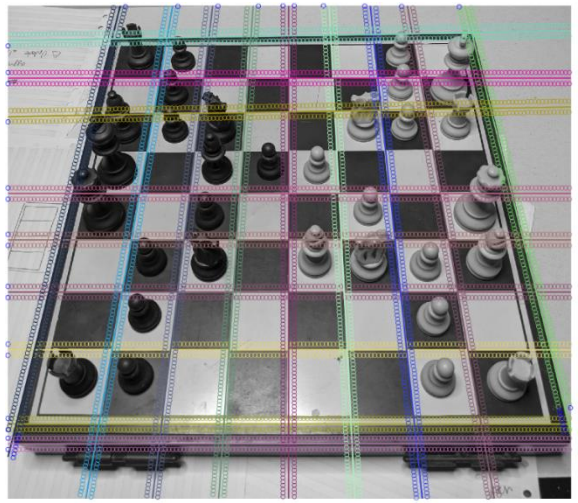


Board location

Houhg lines

Hough lines are used to find the board in the image. As the board has very sharp features, it is easy to find it even if the image background is not completely featureless.

The board is composed of an 8×8 grid, so we look for the 18 most prominent lines in the original image.

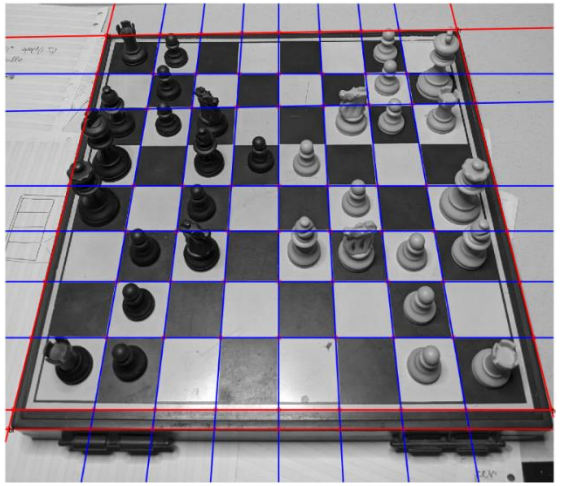


Analysis of alternating gradient property

Alternating gradient property

Lines that divide the board squares internally will have, on the regions near to both sides of the line, a pattern of alternating dark/light sequences. Lines on the side of the board, will instead present this pattern on at most one side.

We select only the latter group of lines hence obtaining the board contour.



Board edge is found by finding the correct group of lines

Piece classification

Grid pieces

Using the same method used to compose the dataset, the board was rectified using information from the board location operation explained before.

This allows to generate samples to be classified by the network.

Process speed-up

To avoid testing for all 64 squares, a heuristic based on the image entropy was used to ignore squares that were likely empty. This speeds up classification because running the heuristic is much faster than running the model for all the empty squares.

A classified piece on the board

Piece colour

Clustering

To distinguish between white and black pieces, k-means (k=2) is used on the mean colour of the central portion of each of the squares that is thought to be populated.

The cluster with the smaller value is relative to pieces with a darker colour and hence pieces belonging to this cluster are assigned to the "black" side, and vice versa.

Inaccuracies

Sometimes, pieces that are in some light conditions or that are covered by other pieces can be misclassified.

Pieces classified by colour

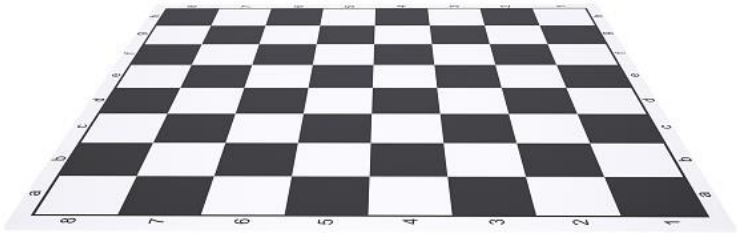
Board orientation

Empty squares

During a chess game, the board orientation is not random; the players face the board from the side where the bottom right square is a white square.

Board orientation can be computed by analyzing the colour of the empty squares.

To gather information about the board alignment, it is sufficient to see if the squares on the even row-column positions are either black or white.



A board picture taken from the side

Board orientation is shown by the yellow arrow

Robustness

To improve performance, the program does not expect all the squares in the required positions to be either black or white, it is sufficient that most than half of the squares be of one colour to determine the orientation.

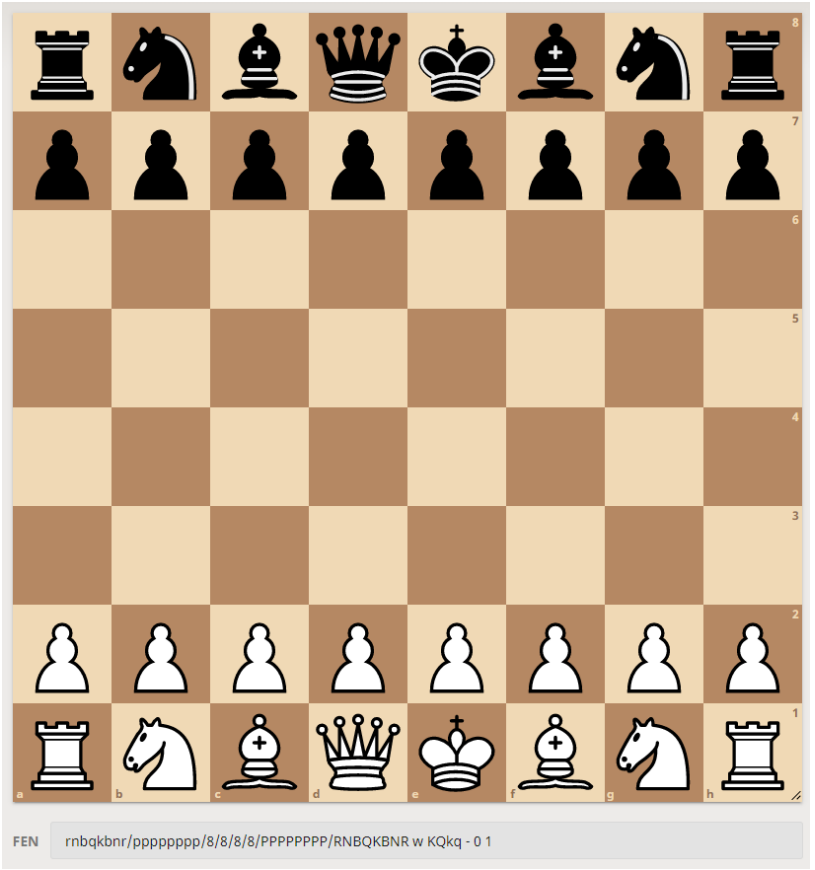
Final output

FEN Notation

The FEN notation gives a representation of the still state of the board and is easily computed after compiling a matrix of each square of the board. The result can be given in string form.

Chess engines

This notation can easily be imported into all major chess engines and from there the position can be analyzed.



The "lichess" interface allows FEN input

Results

Experiments

Experiments have been carried out on a separate set of images, by running the full stack of operations. The only input given to the program is the original image.

Results

Results match the accuracy of the deep learning model, pieces on the board are recognized with very few errors.

Imprecision in the computation of the board contour result in some wrong piece side predictions; in fact, manually selecting the board corners solves this issue.

Final result example

Drawbacks

Board location

Some pieces might be classified incorrectly if the tentative location is not precise enough. This could be improved by using the board location model as hints, allowing the user to then execute small adjustments when needed and provide a more precise board localization

Occlusion

There currently is no way to correctly classify pieces that are occluded by other pieces. It is reasonable to think that there is no way to solve this problem in an accurate way, as no information can obviously be gathered from the image if a piece, say a short pawn, is fully in shade of another, taller piece, say a queen. It is possible that using a chess engine, a prediction could be made, but it would only be a speculation and very player dependent.

Conclusion

Contributions

With this work, we provide a program able to recognize a chessboard starting from a picture taken from a user with an uncalibrated camera phone and no special equipment.

We also provide a new dataset of chess pieces that, even if limited, to the best of our knowledge is larger than the ones publicly available.

Future work

Future directions might include the development of a smartphone app able to perform the entirety of these operations on the mobile device.

The performance of the model and value of this work could be greatly improved by a crowd-sourcing effort aimed at enlarging the dataset to encompass many different chess sets.

Running a demo

Demo instructions

1) Load 'trainedNet' workspace (You only need to do this once)

2.1) Choose a sample image from folders boards_test or boards_test2
2.2) Write the path to the image in the first line of the "run.mlx" file
2.3) If you chose a file from boards_test, set variable "toRotate" to 0, if you chose an image from boards_test2, set it to 1.
This is because MATLAB does not recognize image rotation information in the metadata. Phones use this data when you take a picture with different phone orientation to keep the image consistent with the phone frame of reference.

3) Start the script 'run.mlx'. All intermediate results will be shown in the live script.

Thank You