LORIS GIULIVI MAT: 920992 CP: 10524166

IACV Project Report

Chessboard recognition

Introduction

Studying chess is a complex matter, when playing over-the-board, it is difficult to analyse a position and, in some situations, there is a need to transfer the state of the board to a computer chess engine. This, however, takes a lot of time if done by hand. The aim of this project is to speed up the process using the tools of computer vision and machine learning.

Results yield a very high level of accuracy and show that this method is successful in speeding up the board analysis process, even in cases where some human input may still be required if some pieces are misclassified.

Problem formulation

The aim of the project is to provide an efficient and accurate program able to reconstruct the chessboard state from a picture taken with a smartphone. The resulting program must be able to reconstruct the state of the chessboard from a single image of a chessboard taken in relatively good conditions, and finally provide a useful representation of the state of the game to be used in chess engines.

In this project, the full stack of board recognition is implemented. The user takes a picture of the board so that the image plane is roughly parallel to one of the edges of the board (either from the front or from the side), and with a pitch of roughly 45 degrees with respect to the plane of the board (see *Figure 1*). It is not necessary that the camera be placed in the exact position as described, however, a better placement will result in a more accurate recognition of the board.

After the image is taken, the program will autonomously recognize the board in the image and subsequently the type of all pieces in the board, the pieces side (either black or white), and the board orientation. Finally, a tentative FEN[1] notation is provided, which can be used to import the board state in an external analysis engine.

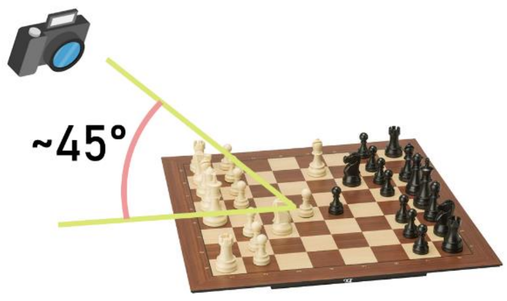


Figure 1 - Camera position during acquisition

It must be noted that due to symmetries in a chessboard, some information may not be gathered. Moreover, castling rights, threefold repetition and en-passant[2] states cannot be captured from the still image of the chessboard.

For example, as shown in *Figure 2*, two very different board configurations can be mistaken one for another. In the example, the only difference in the two boards is the rotation. Without knowing where the white and black player sit at the table, it is impossible to distinguish between the two board states. Without knowledge about the board rotation, we cannot know in which directions the pawns will move. This is especially relevant in the example as the configuration

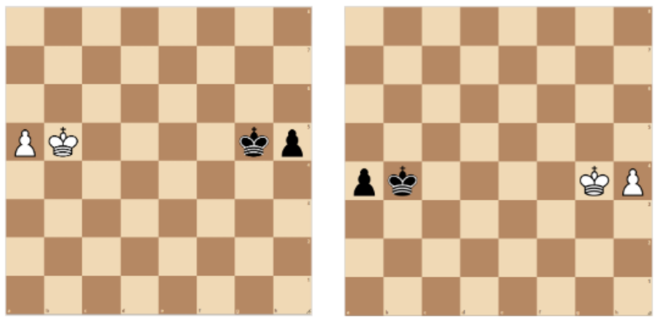


Figure 2 - Rotation equivalent states (white player always on the bottom side)

[1] Forsyth–Edwards Notation (FEN) is a standard notation for describing a particular board position of a chess game. https://en.wikipedia.org/wiki/Forsyth%E2%80%93Edwards_Notation

[2] Rules of chess: Castling, En passant, Threefold repetition. https://en.wikipedia.org/wiki/Rules_of_chess

on the left is winning for white while the one on the right is winning for black.

Related work

Chessboard and chess piece 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.

In "Chessboard and chess piece recognition with the support of neural networks”[3], the 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.

In "ChessVision: Chess Board and Piece Recognition"[4], in a very controlled environment, piece recognition was able to reach 85% accuracy.

In other cases, green/red boards were used instead of the standard white/black, or special 3d printed pieces were used to be shape-matchable.

It is clear that some additional information or simplification of the problem must be performed to solve the problem with current technology. In this work, we want to remove any visual aid from the scene, such as the ones used in similar work, so that the program could be used by anyone with a standard board and pieces, provided that the model is also trained to recognize that specific board and piece set.

Approaches

A general description of the approaches that have been explored is described here, for a more in-depth explanation of the approach that was then used see the *Implementation section*.

Geometry

A purely geometrical approach has been tried at the beginning of the work, but due to the board being very cluttered, it was not possible to find the contour of pieces to perform recognition based on their geometrical features alone. This because there are many cases in which pieces are surrounded by other pieces that have the same colour and texture, therefore no known segmentation algorithm was able to distinguish between the different portions of the image.

Calibration using geometry of the board and vertical vanishing points was considered but it was deemed unfit for the solution for two main reasons: Firstly, we wanted this work to be the basis for a general recognition framework; it is clear that there are some chess sets/environment combinations that make it very difficult to compute a vertical vanishing point, for example a very feature poor environment or a very thin chessboard. The second reason is that even for cases where vanishing point can be computed, it could be a source of error in the location of the board within the image, as many new lines would have to be found in the image, and some of which could be misinterpreted as board lines. For these reasons, and because the used approach is already precise enough in locating the board, this method was discarded.

To achieve the desired results, therefore, only the geometric features of the board plane have been used to pre-process the picture and to locate the board in the image; piece classification has been carried out using other approaches.

Deep learning

A convolutional neural network was used to perform classification of the pieces of the board. Various architectures were tested, but transfer learning on a *resnet50* network gave the best results. To further improve the recognition of pieces, a specific cropping of the input image was used.

Colour

To distinguish between piece colour and board direction using the board squares, k-means (k=2) clustering has been extensively used throughout the project. This allows for independence of specific board/piece colour and light conditions.

Dataset

At the time of writing, no chess piece recognition dataset exists that is large enough to train a neural network. The only available datasets contain less than 100 images per class. To perform training of our model, a dataset was hence created from scratch, by taking pictures of a standard chessboard in a desk

[3] Chessboard and chess piece recognition with the support of neural networks https://arxiv.org/abs/1708.03898

[4] ChessVision: Chess Board and Piece Recognition https://web.stanford.edu/class/cs231a/prev_projects_2016/CS_231A_Final_Report.pdf

environment with some objects around it and using a phone camera (Pixel 3a).

Acquisition

A standard board was randomly populated, and a picture was taken from each side, for a total of 4 pictures for board configuration. The phone was hand-held. In total, 177 board pictures were taken in different light conditions and in different surrounding environments, for a total of 3’629 samples containing pieces and empty squares.

For each of the board images, pieces were extracted using MATLAB. The script that allows to extract the pieces requires the user to select, for each image, the four corners of the board inside the image. From this information the image is rectified with high precision assuming that the board is square, and the board squares can be located by dividing the obtained region into an 8×8 grid.

As the board only contains at most 32 pieces, it is mostly composed of empty squares. As the dataset must be balanced for correct training of the model, it would be useless to acquire a vast amount of empty square samples that would then have to be discarded. To solve this problem, each square from the rectified image was clustered based on the entropy of the relative image portion. Empty squares usually have a low entropy value while populated squares will be more feature rich and have higher entropy; this allows to distinguish between the two types and pre-emptively discard useless data.

If, however, the model was then to analyse empty squares it had not been trained for, the approach of discarding this samples would negatively impact the performance of the model. This means that the process of entropy clustering must not only be done during training, but also when running predictions. In fact, the code that clusters and removes empty squares is used both for dataset acquisition and for chessboard recognition.

Another advantage of this approach is that the model is required to perform less predictions, at the cost of having to run k-means, but because clustering is done on the mean values of the colour, it is one dimensional, and it has to be run only once for the whole board, hence it is much faster than running the deep learning model for each of the empty squares.

This is not to say that the model is not trained to classify empty squares. K-means is not perfect; a fair number of empty square samples may leak into the wrong cluster. It is sufficient, however, to leave in the dataset the squares that were wrongfully clustered during acquisition to have a good representing population of the empty squares that the model will have to classify during chessboard analysis.

After each piece was recognized on the board, a picture crop containing the piece was saved. The crop has the bottom edge aligned and of the same dimension of the square on which the piece is located and is twice as tall. A representation of a piece crop can be seen in *Figure 3*.

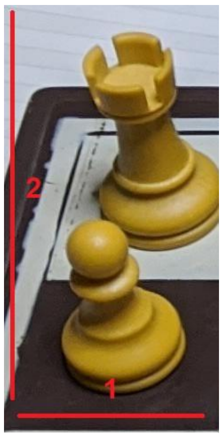


Figure 3 - Piece crop for a pawn

Labelling

To obtain ground-truth for the samples, a custom-built program was developed using *AutoIt*. The program presents a graphical interface (*Figure 4*) that allows with the press of a button to assign a specific piece class to each of the samples that are taken from the acquired dataset and shown one by one. When the button is pressed, the image is copied to the relative folder and its class is thus defined.

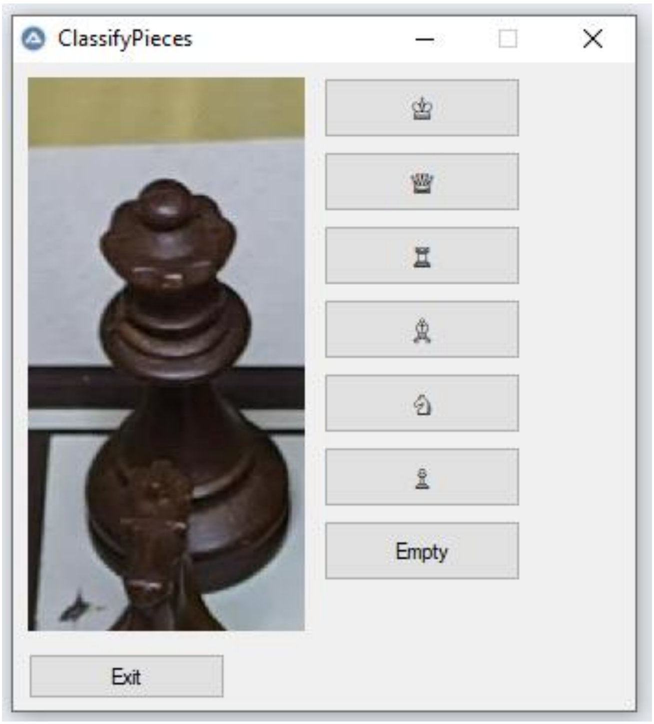


Figure 4 - Interface of the labelling program

The result is a set of 7 folders, each containing indexed images that refer to a specific piece type. Each folder is named accordingly to the type of piece the images it contains represent.

Implementation

Implementation has been carried out using MATLAB on a Windows machine. Additional support software to create the dataset has been developed in AutoIt.

Board location

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. Standard pre-processing steps are taken to create an input image to feed to the Hough transform function. The board is composed of an 8×8 grid, so we look for the 18 highest peaks in the edge image Hough transform, corresponding to the 18 most prominent lines in the original image.

To find lines that belong to the edge of the board, we filter from all the found lines based on the "alternating gradient property" that is more prevalent in lines inside the board. 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 (if the board is very tight, the pattern will be seen on one side only, if the board edges are a bit further from the board grid itself, the pattern will not be seen in any of the sides).

For each line, an average colour is taken for samples on each side of the line at regular intervals, with a fixed kernel. The average colour is then clustered using k-means (k=2) and the property is checked by scanning the clustering results.

In *Figure 5*, we can see the lines that were analysed, together with the regions from which the samples were taken. The colour of the line here is random but is consistent with the colour of the region points. Some blue points at the edges represent samples that could not be acquired because of out-of-bounds regions.

To check for the property, we scan the clustering results along the line. If we have that streaks of samples belong to the same cluster, and that streaks alternate between clusters enough times, then the "alternating gradient property" is verified, otherwise it is not. The property is fully parametric and may require some slight adjustments based on the image resolution and board type.

In a perfect environment, this operation would be useless as taking the outermost lines would suffice in obtaining the board bounds. In some cases, however, artifacts in the image may result in some lines that are wrongfully recognized, as Hough lines sometimes cross the image in completely different directions. This would result in a very imprecise board bound and subsequent piece recognition. With this approach, instead, lines that cross the image at different angles will most likely also cross the board, satisfying the alternating gradient property on both sides and hence being discarded.

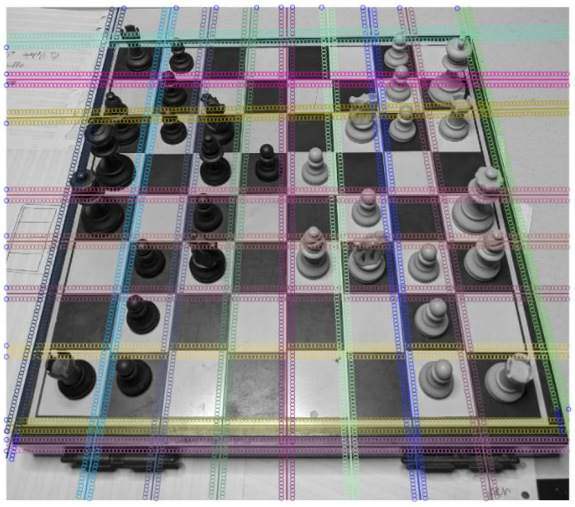


Figure 5 - Analysis of alternating gradient property

The results are very accurate, and provide the required filter needed to obtain the board edges. The board corners can then be computed by intersecting all the lines and taking the outermost intersections.

In some cases, more than four lines are found, and in more extreme cases, such as in *Figure 6*, the corners might not be consistent. In the example figure, the lines that should form the bottom right corner do not intersect in the image, so the outermost bottom right intersection comes from a different pair of lines. The

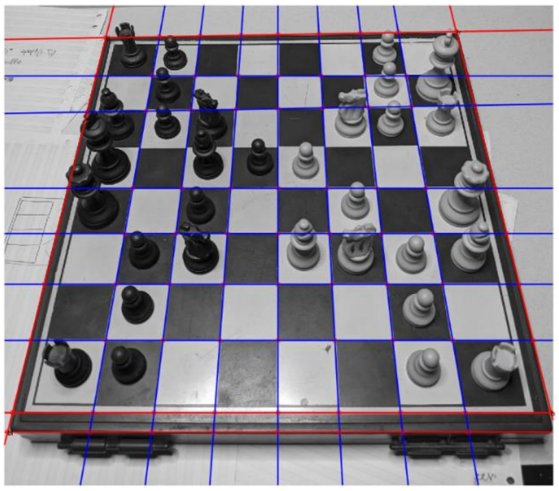


Figure 6 - Non consistent corner recognition

model is however sufficiently robust to still be able to recognize all the pieces.

Piece classification

To classify the pieces by type, the deep learning model is used. The model is trained on an augmented version of the dataset to increase robustness, where each input image is randomly translated and rotated within a small range. The augmentation also consists in a diamond shaped crop that allows the network to focus more on the piece by blacking out other pieces that might be present in neighbouring squares. This crop is also performed during prediction. A sample crop can be seen in *Figure 7*.

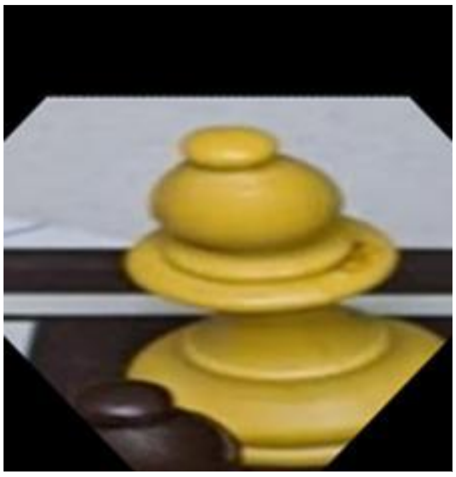


Figure 7 - Sample diamond crop

The model is trained by transfer learning on a *resnet50* network pre-trained on *ImageNet*, where only the classification layer was replaced. The network achieves performance of 96% on the test set (30% of the dataset).

The model receives as input a 224×224 image that is a rescaled and cropped version of the image coming from the original board crop, as described in the *Acquisition section*. During board recognition, the program collects the samples in the same way as during the dataset acquisition, the only difference being that the corner points are not given by the user but are instead the result of the board location process, and may be somewhat imprecise. The board is hence rectified by finding the homography that maps the four corners in the image to four corners of a square. The board grid squares can easily be found by dividing the rectified square board into a grid and mapping back the results onto the original image. Entropy clustering can then be computed to distinguish between populated and empty squares.

Finally, the square aligned 2×1 crop is obtained, then it is further scaled and cropped to match the input format (*Figure* 7) and is fed to the network.

Piece side recognition

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. A depiction of the process can be seen in *Figure 8*.

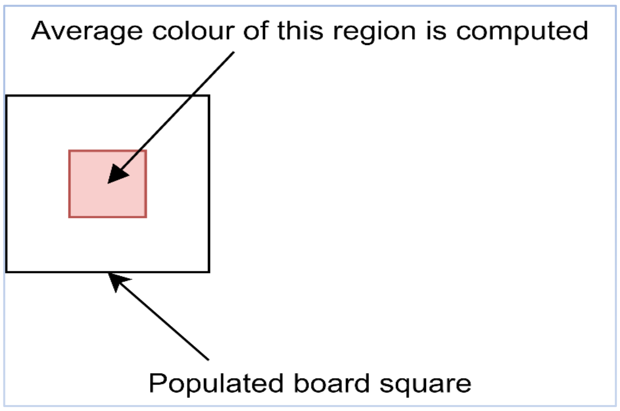


Figure 8 - Piece colour recognition

This is because the piece is expected to be in the central region of the square and is smaller than the full grid square; hence, computing the average colour over the whole square would give inaccurate results.

The two clusters are then compared based on the representative value, that for k-means is the centroid of the cluster. 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.

Sometimes, pieces that are in some particular light conditions, shadows being one of these conditions, can be misclassified.

Board orientation

Board orientation can be computed by analysing the colour of the empty squares. A similar approach to the one used to compute piece colour is used to compute the colour of the non-populated squares, albeit using all the square area for more robustness.

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

As shown in *Figure 9*, if the board picture is taken from the side, the even row-column positions as seen from the input image will be black squares.

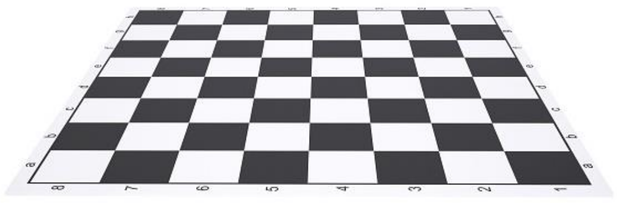


Figure 9 - Picture of chessboard from the side

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.

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 of the board squares that contains information about if and what piece lies on that square. The matrix must be rotated depending on the board direction. The result can be given in string form.

Experiments and results

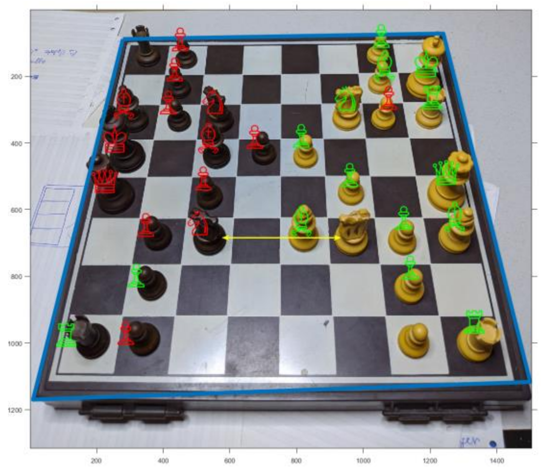
Experiments have been carried out on a separate set of images, by running the full stack of operations. The only input given to the program is the original image. The program outputs the final FEN string, but results can also be visualized by overlaying the predictions over the original image.

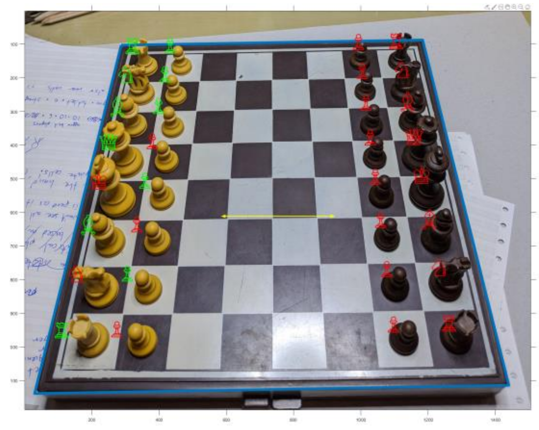
Results match the accuracy of the deep learning model, pieces on the board are recognized with very few errors. Some false negatives (squares with a piece that is mistakenly recognized as empty) and some errors in the piece colour assignment are a result of the slight imprecision in the computation of the board contour; in fact, manually selecting the board corners solves this issue.

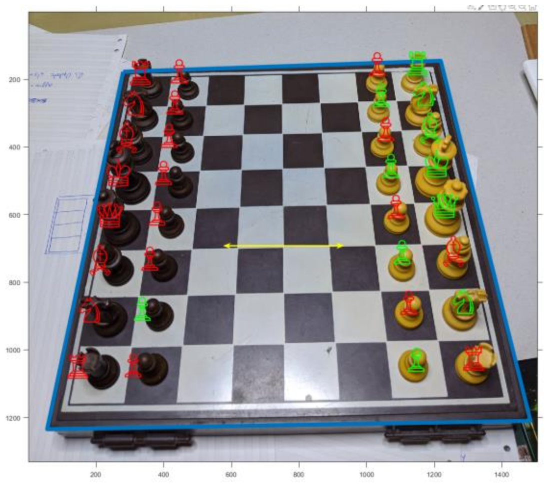
The following are some samples of board recognition, where pieces are marked in their location. Green markers represent white pieces, red marker represent black pieces. The board bounds are represented by a thick light blue line.

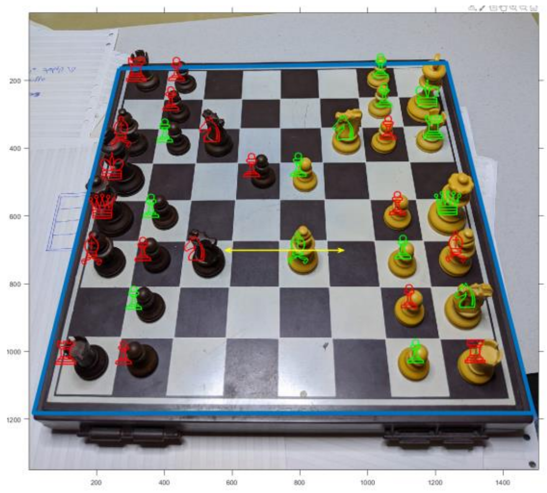
The yellow double arrow represents the board orientation, with the arrow heads pointing towards the players' positions.

The selected samples come from the hardest possible configurations the model had to face, namely, the board state at the beginning of the game, where pieces are most cluttered, and in the opening portion of the game, where all the pieces are still in game and in configurations where they might occlude one another. In the samples we can see how samples were taken at slightly different angles w.r.t. the desk plane, further displaying the model's robustness.









Conclusions and future work

In conclusion, 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.

With this work, we achieved the objective of providing a better way of importing the state of an over-the-board game in a digital format. Even in cases where some pieces might be misclassified, an interface could be provided to fix these errors relatively quickly. This means that this method can be faster than manually copying the board state into an analysis application.

Future directions might include the development of a smartphone app able to perform the entirety of these operations on the mobile device, or with some web application to support the operations. The result of the board state reconstruction could then be directly imported into a chess engine on the user's mobile phone to be analysed, thus accelerating the chess learning process.

Open issues and discussion

Some issues are left open in this work and are here discussed. Firstly, the board location process is subject to noise. Some pieces might be classified differently if the tentative location is not precise enough as the error is then transferred to all the grid squares that will be analysed. 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.

Another problem regards the dataset size. As stated before, the provided dataset is limited. A more general dataset including pieces coming from different sets would greatly increase the value of this work, as the model would then be able to generalize to a wider range of pieces and boards and be usable by many users. A crowd-sourcing effort would be the best way to enlarge the dataset as all the necessary instruments to do so are present in this work, including a user-friendly labelling tool.

Finally, a problem for which no solution is provided, is the one regarding occluded pieces. 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.

APPENDIX A – Other results samples