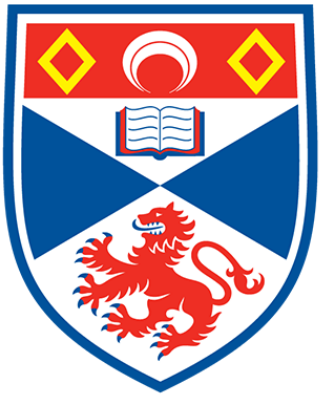
MSci Dissertation



University of
St Andrews

Chess recognition using machine learning

Author:
Georg Wölflein

Supervisor:
Dr. Ognjen Arandjelović

3rd November 2020

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is XX,XXX words long.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Georg Wölflein

Contents

List of Figures

List of Tables

Notation

This report will follow notation conventions established in the deep learning community, in particular those described in Goodfellow *et al.* [1].

...

list symbols, etc.

Chapter 1

Introduction

Former World Chess Champion Garry Kasparov remarks that "improving your weaknesses has the potential for the greatest gains" [2] – a profound observation that may even apply outside of chess. With regard to chess in particular, Kasparov implies that you must identify your mistakes and weaknesses in order to improve as a player, and to do so, you must analyse your own games.

Amateur chess players can analyse games they played online without much effort because the moves are recorded automatically. However, to analyse over-the-board games[1], players must tediously enter the position in the computer piece by piece. A casual over-the-board game between two friends will often reach an interesting position[2]. After the game, the players will want to analyse that position on a computer, so they take a photo of the position. On the computer, they need to drag and drop pieces onto a virtual chessboard until the position matches the one they had on the photograph, and then they must double-check that they did not miss any pieces.

The goal of this project is to develop a system that is able to map a photo of a chess position to a structured format that can be understood by chess engines, such as the widely-used Forsyth–Edwards Notation (FEN) [3], in order to automate this laborious task.

1.1 Context survey

Determining the game state of a chess board, also known as *chess recognition*, is a problem in computer vision whereby an algorithm is tasked with recovering the configuration of pieces from an image of a chessboard. Early work on chess recognition in the 1990s focused on extracting typeset games from printed material [4]. In recent years, the problem of parsing two-dimensional chess images has effectively been solved using conventional machine learning techniques [5] and deep learning [6], [7]. However, recognising chess positions from physical

[1] Usually, players will invest more effort in over-the-board games, both in terms of time and deep thinking. These games will also involve a greater psychological aspect as a result of being able to observe the opponent's expressions. As such, analysing these games should be even more interesting and fruitful.

[2] For example, one of the players might have a few moves that look promising, but is also considering a line with a piece sacrifice. If he decides to play it safe, he will likely want to analyse the piece sacrifice on the computer after the game.

chessboards as opposed to artificial two-dimensional images poses a much more interesting and challenging problem that finds practial application in chess-playing robots, augmented reality, and aiding amateur chess players[3].

Chess robots Initial research into chess recognition emerged from the development of chess robots that included a camera to detect the human opponent's moves from a top-down overhead perspective. The difficulty of distinguishing between chess pieces from a bird's-eye-view due to their similarity is noted in many papers; as a result, chess robots typically implement a three-way classification system that for every square attempts to determine whether it contains a piece, and if so, the piece's colour. Various approaches have been explored including employing manual thresholding [9]–[12] and clustering [13] in different colour spaces, as well as differential imaging (classifying based on the per-pixel difference between two images) [14], [15]. Although the *Gambit* robot proposed by Matuszek *et al.* [16] does not require a bird's-eye view over the chessboard and uses a depth camera to more reliably detect the occupancy of each square, it employs the three-way classification strategy using a linear support vector machine (SVM) to determine the piece colour.

Chess move recording Several techniques for recording chess moves from video footage have been proposed that follow a similar three-way occupancy and colour classification scheme, both from a top-down perspective [8], [17] as well as from a camera positioned at an acute angle to the board [18]. However, in any three-way classification approach, the robot or move recorder requires knowledge of the previous board state in addition to its predictions for each square's occupancy and piece colour to deduce the last move. While this information is readily available to a chess robot or move recording software, this is not the case for a chess recognition system that should deduce the position from a single still image. Furthermore, these approaches experience severe shortcomings in terms of their inability to recover once a single move was predicted incorrectly and fail to identify promoted pieces[4] [9].

Single-image chess recognition A number of techniques have been developed to address the issue of chess recognition from a single image. Unlike move recording software or chess robots, it does not suffice to only determine the occupancy and colour of each square, but each piece must be identified. These techniques must implement a classification algorithm for each piece type (pawn, knight, bishop, queen, and king) of each colour which poses a significantly more difficult problem, attracting research mainly in the last five years. From a bird's-eye view, the pieces are nearly indestinguishable, so the photo is usually taken at an angle to the board. Ding [19] proposes a piece classifier that uses one-versus-rest SVMs trained on scale-invariant feature transform (SIFT) and histogram of oriented gradients (HOG) feature descriptors, achieving an accuracy of 85%. Danner and Kafafy [20] as well as Xie *et al.* [21] claim that SIFT and

[3]Electronic chess sets are impractical and very costly [8], thus solutions for chess recognition using just a photo of an unmodified chess board are more compelling for amateur chess players.

[4]Piece promotion occurs when a pawn reaches the last rank, in which case the player must choose to promote to a queen, rook, bishop or knight. Evidently, a vision system that can only detect the piece's colour is unable to detect what it was promoted to.

HOG provide inadequate features for the problem of piece classification due to the similarity in texture between chess pieces, and instead focus on the pieces' outlines. As such, Danner and Kafafy [20] use Fourier descriptors calculated for the pieces' contours, but this requires a manually-created database of piece silhouettes. Furthermore, they modify the board colours to red and green instead of black and white, in order distinguish the pieces from the board more easily[5]. On the other hand, Xie *et al.* [21] perform contour-based template matching with an interesting caveat: the camera angle is calculated based on the perspective transformation of the chessboard, and then depending on the angle, different templates are utilised for matching the chess pieces. As part of the same work, Xie *et al.* developed another approach that instead utilised convolutional neural networks (CNNs), but found that their original template-matching technique achieved superior results in terms of speed and accuracy in low-resolution images. However, it is important to note that their CNNs were trained on only 40 images per class and deep learning methods tend to excel when trained on larger datasets.

Chessboard detection A prerequisite to any chess recognition system is the ability to detect the location of the chessboard and each of the 64 squares. Once the four corner points have been established, finding the squares is trivial for pictures captured in bird's-eye view, and only a matter of a simple perspective transformation in the case of other camera positions. While finding the corner points of a chessboard is frequently used for automatic camera calibration due to the regular nature of the chessboard pattern [22], [23], techniques designed for this purpose tend to perform poorly when there are pieces on the chessboard that occlude lines or corners. Some of the aforementioned chess robots [13], [14], [17] as well as the single-image recognition system proposed by Danner and Kafafy [20] circumvent this problem entirely by prompting the user to interactively select the four corner points, but ideally a chess recognition system should be able to parse the position on the board without human intervention. Most approaches for automatic chess grid detection utilise either the Harris corner detector [11], [18] or a form of line detector based on the Hough transform [12], [15], [20], [24]–[27], although other techniques such as template matching [16] and flood fill [8] have been explored. In general, corner-based algorithms are unable to accurately detect grid corners when they are occluded by pieces, thus line-based detection algorithms appear to be the favoured solution. Such algorithms often take advantage of the geometric nature of the chessboard which allows to compute a perspective transformation of the grid lines that best matches the detected lines [18], [21], [24]. However, lines found in the background of the photo can often cause failure modes. A recent chess grid detection algorithm that is highly successful even on populated boards is descibed by Xie *et al.* in [27]. They apply several clustering algorithms on the lines detected via a Hough transform in order to find the horizontal and vertical grid lines belonging to the chessboard, and use this algorithm as a preprocessing step in their template-matching piece classification technique [21] described above.

[5]Similar board modifications have also been proposed as part of chess robots [11] and chess move trackers [8], but any such modification imposes an unreasonable constraint on normal chess games.

Chess recognition using CNNs Since Xie *et al.* pioneered the use of CNNs in the domain of chess recognition from monocular images in 2018[6], a few more techniques have been developed that employ CNNs at various stages in the recognition pipeline. Czyzewski *et al.* [29] achieve an accuracy of 95% on chessboard detection from non-vertical camera angles by designing an iterative algorithm that gereates heatmaps over the input image representing the likelihood of each pixel being part of the chessboard. They then employ a CNN to refine the corner points that were found using the heatmap, outperforming the results obtained by Gonçalves *et al.* [13]. Furthermore, they compare a CNN-based piece classification algorithm to the SVM-based solution proposed by Ding [19] and find no notable improvement, but manage to obtain major improvements by implementing a probabilistic reasoning system that uses the open source Stockfish chess engine [30] as well as chess statistics [31]. Although reasoning techniques were already employed for refining the predictions of chess recognition systems before [20], [25], Czyzewski *et al.* demonstrate the potential of combining information obtained from a chess engine with large-scale chess statistics. Very recently, Mehta and Mehta [32] implemented an augmented reality app using the popular *AlexNet* CNN architecture introduced by Krizhevsky *et al.* [33], achieving promising results. Despite using an overhead camera perspective and not performing any techniques to ensure probable and legal chess positions, Mehta and Mehta achieve an end-to-end accuracy of 93% for the entire chessboard detection and piece classification pipeline.

Datasets The lack of adequate datasets for chess recognition has been recognised by many [19], [29], [32]. Although Czyzewski *et al.* [29] published a dataset of chessboard lattice points that are difficult to predict [34], large datasets – especially at the scale required for deep learning – are not available as of now. Using synthesised data in the training set is an efficient means of creating sizable datasets while minimising the manual annotation efforts [28], [29], [35]. Czyzewski *et al.* distort some input images in order to simulate different camera perspectives on the chessboard corners. However, a more promising method seems to be the use of three-dimensional models. Wei *et al.* [28] synthesise point cloud data for their volumetric CNN directly from three-dimensional chess models and Hou [35] use renderings of three-dimensional models as input. Yet Wei *et al.* [28]'s approach works only if the chessboard was captured with a depth camera and Hou [35] presents a chessboard recognition system using a simple artificial neural network (ANN) that is not convolutional and hence achieves an accuracy of only 72%.

1.2 Objectives

1.2.1 Primary

1. Perform a literature review of available methods for parsing chess positions from photos.

[6]Wei *et al.* [28] developed a chess recognition system using a volumetric CNN one year previously, but this approach requires three-dimensional chessboard data obtained from a depth camera. Their approach achieved a per-class accuracy over 90% except for the "king" class, was trained on computer-aided design (CAD) models, and evaluated on real three-dimensional images (point clouds) of a chessboard.

2. Develop an algorithm for detecting the corners of the chessboard as well as the squares.
3. Develop an algorithm for recognising the chess pieces.
4. Develop an algorithm that uses the outputs from (2) and (3) in order to compute a probability distribution over each piece in each square.
5. Evaluate the performance of the developed algorithms.

1.2.2 Secondary

1. Create a large labelled dataset of synthesised chessboard images using 3D models.
2. Implement an algorithm that takes as input the raw probability distribution of each piece in each square and outputs a likely Forsyth–Edwards Notation [3] (FEN) description.
3. Implement a simple web API that performs the inference pipeline for an input image, returning the FEN description.

1.2.3 Tertiary

1. Develop a web app that allows the user to upload an image of the chess board to obtain the FEN description.

1.3 Ethics

There are no ethical issues raised by this project, as indicated in the signed ethics form in appendix B.

Chapter 2

Background

2.1 Convolutional neural networks (CNNs)

2.2 Transfer learning

Chapter 3

Design

3.1 Overview

3.2 Data synthesis

Studies in human cognition by Bilalić *et al.* [36] and Zhou [37] compared skilled chess players with novices in terms of their ability to remember a chess position for a short amount time confirmed that highly skilled players outperform novices at this task. Perhaps more interestingly, both studies found that skilled players remembered *random* positions (where pieces are positioned on random squares, not necessarily obeying the rules of chess) significantly less accurately than they did positions from actual chess games. Thus it stands to reason that in general, highly skilled chess players exhibit a more developed pattern recognition ability for chess positions than novices, but this ability is specific to positions that conform to the rules of chess and are likely to occur in actual games.

This project aims to develop a similar pattern recognition ability using machine learning, and therefore our dataset will consist of positions from real chess games. In doing so, we automatically ensure that the chess positions are legal according to chess rules such that a probabilistic reasoning system could later yield sensible results in a post-processing step (see Objective 2.2).

3.2.1 Chess positions

The positions are generated from a publicly available dataset of 2,851 games played by current World Chess Champion Magnus Carlsen [38]. Each move in each game is included in our dataset with a probability of 2%, and duplicate positions are discarded. A total of 4,888 chess positions are obtained in this manner and saved in FEN format.

3.2.2 Three-dimensional renders

In order to obtain realistic images of these chess positions, we employ a three-dimensional model of a chess set on a wooden table. Chess pieces are placed on the board squares according the given FEN description. Different camera angles and lighting setups are chosen in a random process in order to maximise diversity in the dataset.

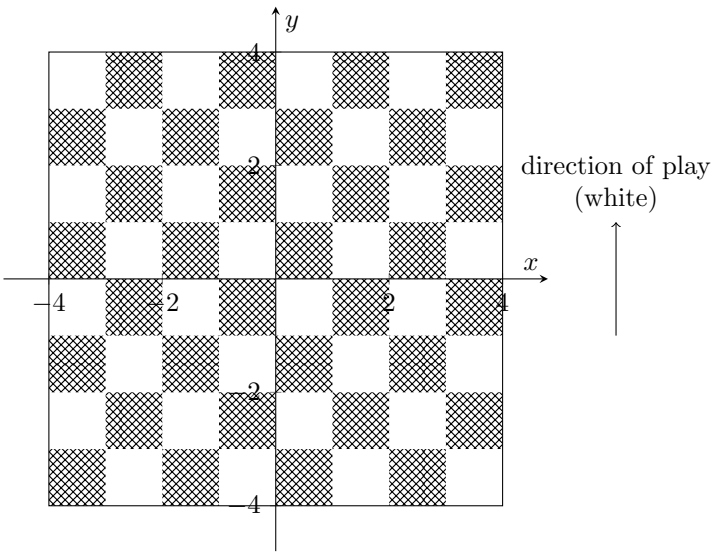


Figure 3.1: Overhead view of the coordinate system on the chessboard. The z-axis (not shown) points upward, normal to the chessboard surface, and the board is oriented like a chess game would be set up, i.e. the bottom right square is white. White's direction of play (the direction in which pawns are advanced) coincides with the y-axis.

Let us consider a three-dimensional Cartesian coordinate system whose origin lies at the centre point of the chessboard's surface, as depicted in fig. 3.1. The chessboard lies on the plane formed by the x and y axes, and the chess squares are of unit length.

Pieces The pieces are positioned on the squares as dictated by the particular FEN description. However, instead of positioning them at the centre in their respective squares, they are randomly rotated and positioned with a random offset to emulate the conditions in real chess games. More specifically, the x and y position of a piece in file i and rank j is sampled from a bivariate normal distribution given by

$$\mathbf{p}_{i,j} \sim \mathcal{N}\left(\begin{bmatrix} i \\ j \end{bmatrix} - \frac{7}{2}, \frac{\boldsymbol{I}_2}{10}\right).$$

Here, we assume that i and j are zero-indexed, i.e. the square *a1* corresponds to $i = j = 0$. The reason for shifting the mean by $\frac{7}{2}$ above is that since the origin lies at the midpoint of the board, the mean must be shifted four units to the left (or downwards for the y-axis), but since the normal distribution should be centred on the midpoint of the square, we must add one half. Due to the fact that the x and y axes are perpendicular, the two components of $\mathbf{p}$ will be independent and thus can be modelled with a covariance matrix that is a multiple of the identity matrix $\boldsymbol{I}_2$. Experiments showed that a variance of $\frac{1}{10}$ achieved realistic results. Finally, the piece's rotation about its z-axis is sampled from a uniform distribution over the half-open interval $[0, 2\pi)$.

Camera The camera is aligned such that it points directly at the origin (i.e. the centre of the board). It is positioned with only a small offset from the

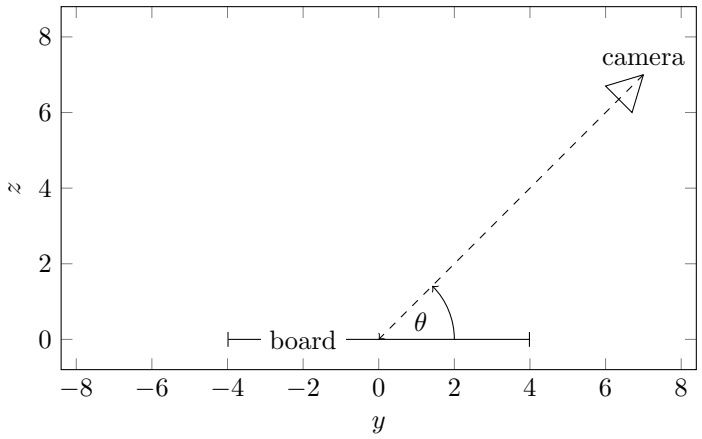


Figure 3.2: Side view of the camera setup for the scenario where it is white to move.

yz-plane to ensure that the view over the chessboard is similar to the current player's perspective. A slight perturbation to the x-component of the camera position is introduced according to a normal distribution with $\mu = 0$ and $\sigma = 0.8$ since the player will not usually be positioned exactly in the middle in front of the board. An angle θ is chosen uniformly in the range $\left[\frac{\pi}{4}, \frac{\pi}{3}\right]$ to represent the angle that the camera makes with the board's surface (see fig. 3.2) if white is to play[1]. This range is chosen because human players would typically choose a camera angle between 45 and 60 degrees to ensure maximum visibility of the pieces. The two remaining components (y and z) of the camera's location is then obtained using a simple trigonometric calculation such that the distance from the camera to the origin is 11 units, a length that allows the camera to capture the entire board.

Lighting For each chess position, a random choice is made between two different lighting scenarios, each having equal probability of being employed.

1. The first lighting mode tries to emulate a *camera flash*. To do so, a spotlight is set up with the same location and orientation as the camera. As a result, the scene is light up quite well with no large shadows, as it can be seen in fig. 3.3(a).

2. In the other lighting mode, two spotlights are set up in the scene. Their x and y coordinates are constrained such that they lie on a circle centred at the origin of the coordinate system with radius 10 on the xy-plane, as depicted in fig. 3.4, but each spotlight's location along the circumference is sampled uniformly. Furthermore, each spotlight's z-component is sampled uniformly in the range $[5, 10)$. Finally, the for each spotlight, a focus point on the chessboard surface (i.e. the xy plane) is sampled from

$$\mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \frac{5}{2}\boldsymbol{I}_2\right)$$

[1]On the other hand, if it is black to play, the perspective must be from the other side of the board, so θ is chosen in the range $\left[\frac{2\pi}{3}, \frac{3\pi}{4}\right]$ which is equivalent to reflecting the camera position about the xz-plane.

(a) camera flash

(b) spotlights

Figure 3.3: Two samples from the synthesised dataset showing both types of lighting.

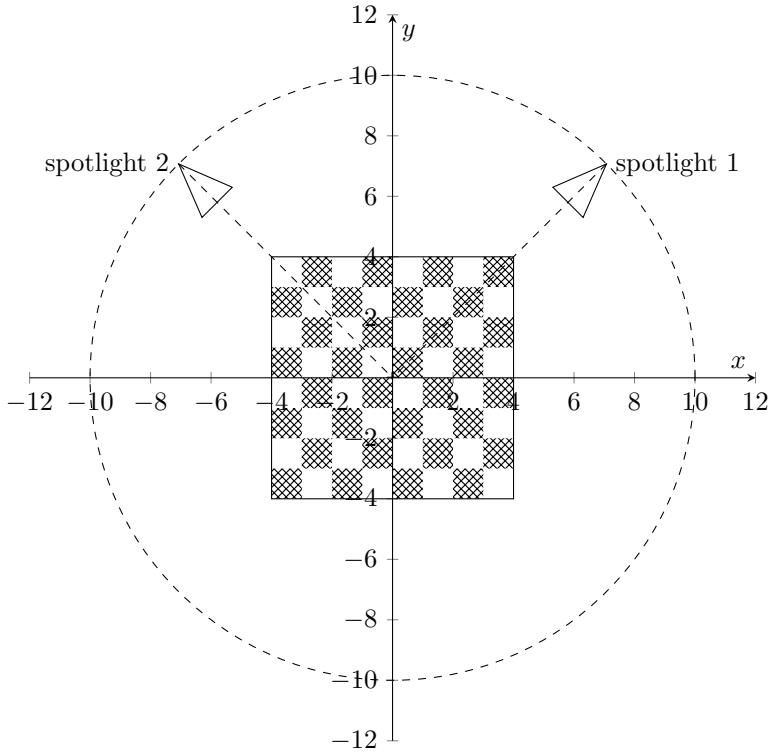
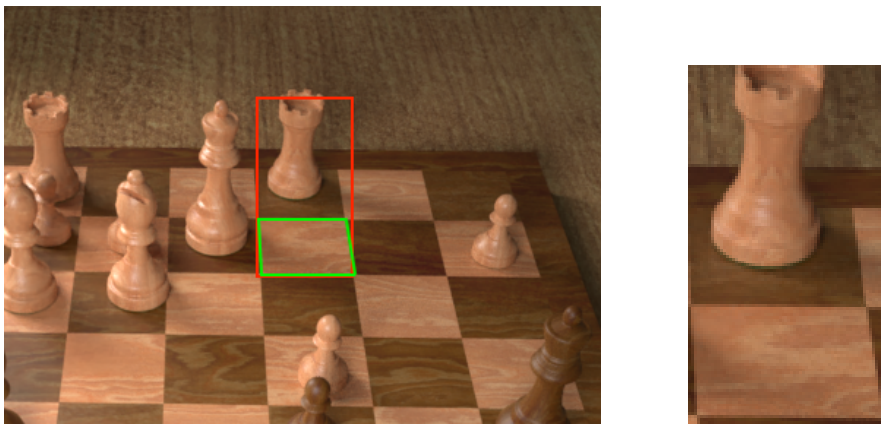


Figure 3.4: Overhead view of the chessboard with two spotlights. The spotlights are constrained to the dashed circle such that their distance to the origin amounts to 10 units when disregarding the z-component.



(a) region from the original image with marked square

(b) cropped sample

Figure 3.5: An example illustraing why an immediate piece classification approach is inclined to reporting false positives. Consider the square marked in green in the original image (a). The bounding box for piece classification (marked in red) must be quite tall because the square might contain a tall piece such as a queen or king (the box must be at least as tall as the the queen in the adjacent square on the left). The resulting sample, depicted in (b), contains almost the entire rook of the square behind. Thus, a piece classifier might classify this square as containing a rook instead of being empty.

and the corresponding spotlight is rotated such that it points in that direction. Consequently, there is greater variabililty in the lighting because the spotlights could be pointing at different areas of the board, thus producing different types of shadows. Figure 3.3(b) shows an example rendering where the lighting produced by the spotlights is poorer than the camera flash mode.

3.2.3 Automated labelling

use the chesscog.data_synthesis.visu script

3.3 A classification-based approach

3.3.1 Board detection

todo: explain how dataset was split into train/val/test

3.3.2 Occupancy classification

TODO: corner point detection

Empirical experiments showed that performing piece classification directly after detecting the four corner points with no intermediate step yields a large number of false positives, i.e. empty squares being classified as containing a chess piece. One common scenario where the trained classifier failed is illustrated in fig. 3.5. Notice that squares further away from the camera must be cropped with increasingly taller bounding boxes. If a particular square is empty but its bounding box includes the piece from the adjacent square as in fig. 3.5(b), the trained classifier was inclined to report a false positive.

To solve this problem, a binary classifier is trained on cropped squares to decide whether they are empty or not. Before cutting out the squares from the original image, the input image is warped to a two-dimensional overhead view by means of a projective transformation. This ensures that all squares are of equal size and that the corners form right angles (which is not the case in the original image due to perspective distortion). To this end, we compute the *homography matrix* $\boldsymbol{H} \in \mathbb{R}^{3\times 3}$ [39] mapping any point $\boldsymbol{p}$ from the original image to the corresponding point $\boldsymbol{p'}$ in the warped image. To simplify notation, we shall consider 2D homogenous coordinate vectors, i.e. three-component vectors with the last component being 1, so the vector $\begin{bmatrix} x & y & 1 \end{bmatrix}^\top$ would represent the point (x, y) in the Cartesian coordinate system. Using this notation, $\boldsymbol{H}$ maps $\boldsymbol{p}$ to $\boldsymbol{p'}$ using the relation

$$\boldsymbol{H}\boldsymbol{p} = s\boldsymbol{p'} \tag{3.1}$$

up to a scalar scale factor s.

Let the 4×3 matrix $\boldsymbol{P}$ contain the pixel coordinates of the four chessboard corners starting in the top left in clockwise order and $\boldsymbol{P'}$ be a matrix of the same size containing the corner coordinates of a square with side length l pixels given by

$$\boldsymbol{P'} = \begin{bmatrix} 0 & l & l & 0 \\ 0 & 0 & l & l \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Notice that $\boldsymbol{P'}$ describes a square whose top left corner is at the origin, and the coordinates are given in the same order as in $\boldsymbol{P}$. Since the homography matrix should map points from the original image to the output image, we obtain from eq. (3.1) the relation

$$\boldsymbol{H}\boldsymbol{P} = \boldsymbol{P'}(\boldsymbol{I}_4\boldsymbol{s}) \tag{3.2}$$

where the vector $\boldsymbol{s} \in \mathbb{R}^4$ represents the scale factor for each point. Equation (3.2) describes an overdetermined set of linear equations, so it is likely that there is no exact solution for $\boldsymbol{H}$. However, we approximate $\boldsymbol{H}$ by finding the solution with the least squared error. Finally, we can use $\boldsymbol{H}$ to apply the projective transformation to the original image such as depicted in fig. 3.6(a), obtaining the warped image in fig. 3.6(b).

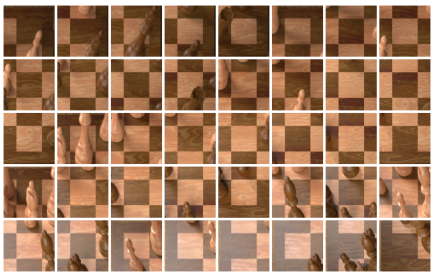
Cropping the squares from the warped image is trivial because the squares are of equal size. In training the occupancy classifiers, it is conjectured that it would be useful to include contextual information with each square; therefore, the squares are not cropped tightly around their boundaries but instead with a 50% increase in length on all four sides, as shown in figs. 3.6(c) and 3.6(d). This might aid the classifier's decision in difficult situations where a chess piece from another square reaches into the cropped one due to the camera perspective.

3.3.2.1 Training CNNs

Six CNN architectures are devised for the occupancy classification task, of which two accept 100×100 pixel input images and the remaining four require the images to be of size 50×50 pixels. They differ in the number of convolution layers, pooling layers, and fully connected layers. When referring to these models, we shall use a 4-tuple consisting of the input side length and the three aforementioned criteria. Figure 3.7 depicts the architecture of the CNN

(a) original

(b) warped



(c) all 32 empty samples

(d) all 24 occupied samples

Figure 3.6: The process of obtaining samples for occupancy classification from a chessboard image. First, the original image (a) is warped to a two-dimensional overhead view, (b). Then, all squares are cropped (with a 50% increase in width and height to include contextual information). Finally, the cropped squares are annoted using the FEN groundtruth as either empty (c) or occupied (d).

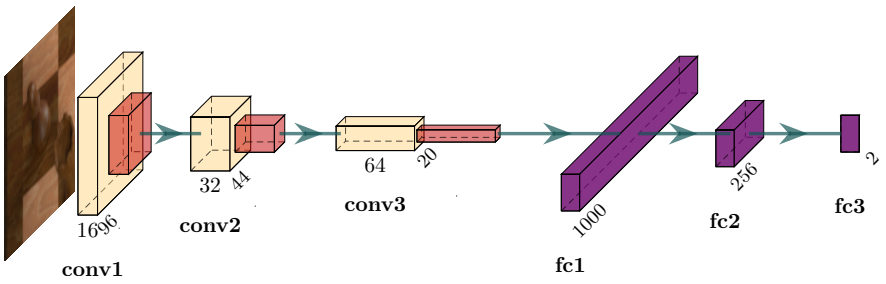


Figure 3.7: Architecture of the CNN $(100, 3, 3, 3)$ network for occupancy classification. The input is a three-channel red, green, blue (RGB) image with 100×100 pixels. There are two convolutional layers (yellow) with a kernal size of 5×5 and stride 1, meaning that each convolutional layer reduces the width and height by 4. The final convolutional layer has a kernel size of 3×3, thus only reducing the input size by two. Starting with 16 filters in the first convolutional layer, the number of channels is doubled in each subsequent layer, as is common practice in CNNs [40]. Each convolutional layer uses the rectified linear unit (ReLU) activation function and is followed by a max pooling layer with a 2×2 kernel that is moved with a stride of 2 such that the width and height are halved. Finally, the output of the last pooling layer is reshaped to a 640,000-dimensional vector that passes through two fully connected ReLU-activated layers before reaching the final fully connected layer with softmax activation.

$(100, 3, 3, 3)$ model which achieves the greatest validation accuracy of these six models. The final fully connected layer in each model contains two output units that represent the two classes (occupied and empty). The models are trained using the cross-entropy loss function on the outputs. Training proceeds using the popular *Adam* optimizer [41] with a learning rate of 0.001 for three whole passes over the training set using a batch size of 128. After every 100 steps, the model's loss and accuracy is computed over the entire validation set, the results of which are reported in fig. 3.8. The model converges smoothly to a very low loss value, achieving a training accuracy of 99.70% and validation accuracy of 99.71%. Due to the fact that the difference between training and validation accuracy is very small (in fact, the validation accuracy even happens to be slightly above the training accuracy), we conclude that the model does not overfit the training set.

3.3.2.2 Transfer learning on deeper models

VGG [40] ResNet [42] AlexNet [33] ImageNet [43]

todo: explain how this was done and explain tradeoff (significantly more parameters in model)

3.3.2.3 Analysis

Each model is trained separately on the dataset of squares that are cropped to include contextual information (by increasing the bounding box by 50% in each direction), and the same samples except that the squares are cropped tightly. In each case, the model trained on the samples that contained contexual information outperfomed its counterpart trained on tightly cropped samples, confirming the hypothesis that the information around the square itself is useful.

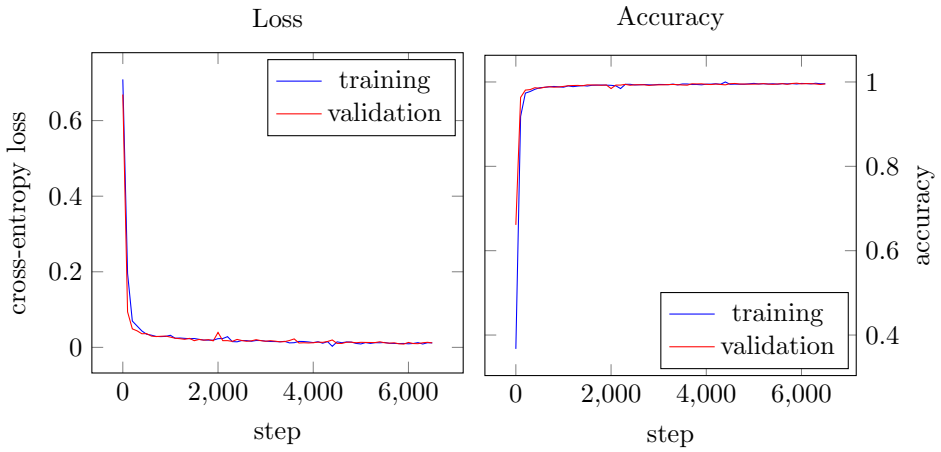


Figure 3.8: Loss and accuracy during training on both the training and validation sets for the CNN $(100, 3, 3, 3)$ model. The best validation accuracy is 99.71%.

	model	parameters	accuracy	precision	recall	errors	training accuracy
✓	ResNet [42]	$1.12 \cdot 10^7$	99.96%	1.000	0.999	4	99.93%
✓	VGG [40]	$1.29 \cdot 10^8$	99.95%	0.999	0.999	5	99.96%
✗	VGG [40]	$1.29 \cdot 10^8$	99.94%	0.999	0.999	6	99.93%
✗	ResNet [42]	$1.12 \cdot 10^7$	99.90%	0.999	0.998	9	99.94%
✓	AlexNet [33]	$5.7 \cdot 10^7$	99.80%	0.998	0.996	19	99.74%
✗	AlexNet [33]	$5.7 \cdot 10^7$	99.76%	0.998	0.995	22	99.76%
✓	CNN $(100, 3, 3, 3)$	$6.69 \cdot 10^6$	99.71%	0.997	0.995	27	99.70%
✓	CNN $(100, 3, 3, 2)$	$6.44 \cdot 10^6$	99.70%	0.996	0.995	28	99.70%
✗	CNN $(100, 3, 3, 2)$	$6.44 \cdot 10^6$	99.64%	0.996	0.993	34	99.61%
✓	CNN $(50, 2, 2, 3)$	$4.13 \cdot 10^6$	99.59%	0.993	0.995	38	99.62%
✓	CNN $(50, 3, 1, 2)$	$1.86 \cdot 10^7$	99.56%	0.997	0.991	41	99.67%
✓	CNN $(50, 3, 1, 3)$	$1.88 \cdot 10^7$	99.56%	0.993	0.994	41	99.66%
✓	CNN $(50, 2, 2, 2)$	$3.88 \cdot 10^6$	99.54%	0.993	0.994	43	99.64%
✗	CNN $(50, 2, 2, 3)$	$4.13 \cdot 10^6$	99.52%	0.993	0.993	45	99.57%
✗	CNN $(100, 3, 3, 3)$	$6.69 \cdot 10^6$	99.50%	0.988	0.997	47	99.55%
✗	CNN $(50, 3, 1, 2)$	$1.86 \cdot 10^7$	99.50%	0.993	0.992	47	99.44%
✗	CNN $(50, 2, 2, 2)$	$3.88 \cdot 10^6$	99.44%	0.995	0.989	52	99.54%
✗	CNN $(50, 3, 1, 3)$	$1.88 \cdot 10^7$	99.39%	0.993	0.989	57	99.41%

Table 3.1: Performance of all occupancy classification models on the validation set. For the CNN models, the 4-tuple denotes the length of the square input size in pixels, the number of convolution layers, the number of pooling layers, and the number of fully connected layers. The check mark in the left column indicates whether the input samples contained contextual information (cropped to include part of the adjacent squares). In the penultimate column, the total number of misclassifications in the validation set are reported (the validation set consists of 9,346 samples). The training accuracy is given in the rightmost column for comparison. Notice that there is no significant difference between the validation and training accuracies, indicating that none of the models suffer from overfitting.

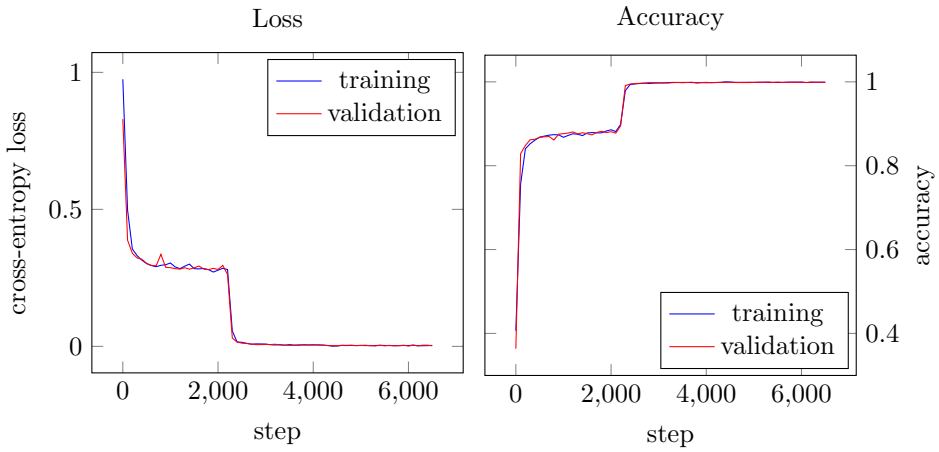


Figure 3.9: Loss and accuracy during training on both the training and validation sets for the ResNet model. The best validation accuracy is 99.96%.

Figure 3.10: The four samples that the ResNet model misclassified in the validation set. The left sample depicts an empty square, and the remaining three are of occupied squares.

3.3.3 Piece classification

Now that the occupancy of each square on the chessboard can be detected to a high degree of accuracy, the next step is to classify the piece in each of the occupied squares. We require a 12-way classifier that takes as input a cropped image of an occupied square and will output the chess piece on that square. There are six types of chess pieces (pawn, knight, bishop, rook, queen, and king), and each piece can either be white or black in colour, thus there are 12 classes of pieces.

We must pay some special attention to how the pieces are cropped. Simply following the approach described in section 3.3.2 for cropping squares does not give enough information to classify pieces. Especially tall pieces at the back of the board would be cropped in a manner such that only the bottom part of the piece remains in the region of interest (ROI). Consider for example the white king in fig. 3.6. Cropping only the square it is located in would not include its crown which is an important feature needed to distinguish between kings and queens. Instead, we must choose a rectangular bounding box that is tall enough to account for the perspective distortion.

The camera perspective causes another phenomenon that we must account for: pieces on the left of the board tend to 'slant' to the left, and vice-versa on the right. This is due to the vanishing point of the normal vectors on the chessboard surface being roughly centred horizontally with regards to the location of the chessboard itself[2], as illustrated in fig. 3.11. Hence, we must extend the ROI vertically in the appropriate direction.

To obtain the ROIs, we first warp the input image of the chessboard as described in section 3.3.2 and exemplified in fig. 3.6(b). At first, each pieces bounding box will correspond to the square it is located on, i.e. its width and height will be that of the square[3]. Depending on the rank r, the height is increased by

$$h_{\text{inc}}(r) = \frac{2r + 5}{7}$$

where the unit of measurement is the height of the chessboard squares. This represents an arithmetic progression such that the bounding box for pieces in the first rank (bottom row of the chessboard, i.e. $r = 1$) will be increased by one square, and pieces in the top row ($r = 8$) will have their height increased by three units.

The increase in width is dependent on the file f and given by the piecewise-defined arithmetic progression

$$w_{\text{inc}}(f) = \begin{cases} -\frac{f}{4} & f \leq 4 \\ \frac{f-4}{4} & f > 4. \end{cases}$$

A negative increase in width means that the bounding box is extended to the left, whereas a positive increase means it is widened to the right. Thus, pieces on

[2]Unfortunately, it is not possible to compute the vanishing point of the normals based solely on the information available in the input images as this would require knowledge of the intrinsic and extrinsic camera parameters [44]. The aim of this work is to recognise chess positions from just the input image without any further information; therefore, we devise a heuristic based on the observation that the vanishing point tends to be vertically below the chessboard and roughly horizontally centred.

[3]Note that due to the warped perspective, all squares have equal width and height.

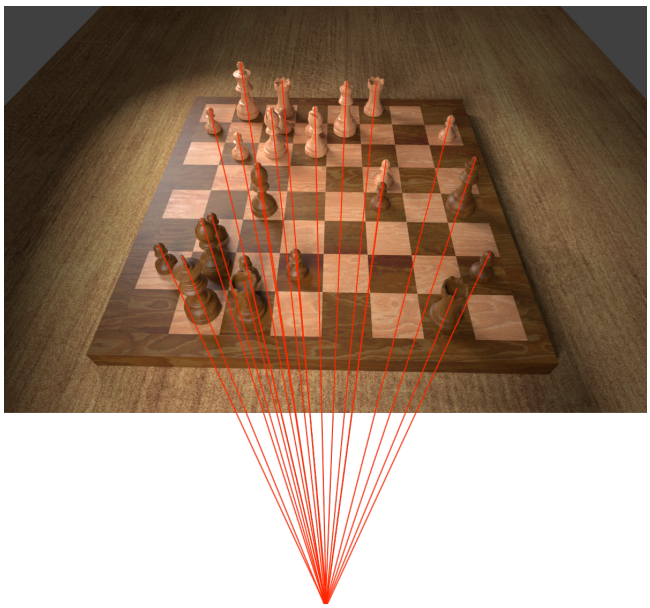


Figure 3.11: The normals of the chessboard surface (corresponding to the direction the pieces are pointing, marked in red) converge to a single vanishing point which is below the image. We will assume that the vanishing point is roughly horizontally centred with the chessboard, as this corresponds to how chessboards are usually photographed. As a result, pieces on left 'lean' left, and vice-versa on the right.

the left half of the board ($1 \leq f \leq 4$) will have their bounding boxes extended to the left, and vice-versa on the right (for $4 < f \leq 8$).

Experiments demonstrate that this heuristic generates bounding boxes that are large enough to contain even the tallest pieces in most cases, whilst not being needlessly large. As a further preprocessing step, the cropped ROIs for all pieces on the left side of the board ($1 \leq f \leq 4$) are flipped vertically, such that the square that the piece stands on will always be in the bottom left of the image. This may help the classifier understand what piece is being referred to in samples where the larger bounding box includes adjacent pieces in the image. Figure 3.12 shows a random selection of the ROIs corresponding to white queens in order to demonstrate that the bounding boxes are large enough to contain even tall pieces.

3.3.3.1 Training CNNs

Similar to section 3.3.2, we train several models on

talk about change in number of epochs etc

3.4 An approach based on object detection

3.5 Identifying plausible game states

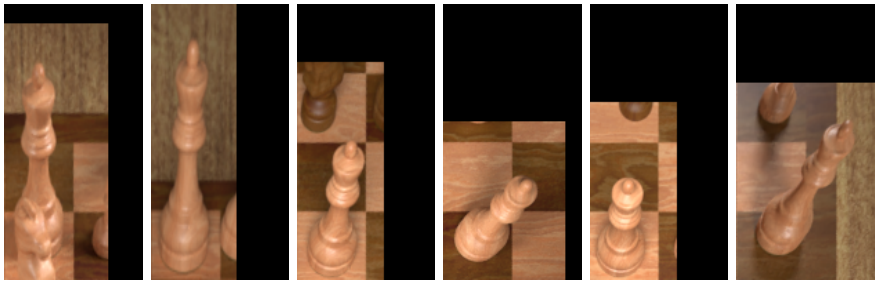


Figure 3.12: A random selection of six samples of white queens in the training set. Notice that the square each queen is located on is always in the bottom left of the image and of uniform dimensions across all samples.

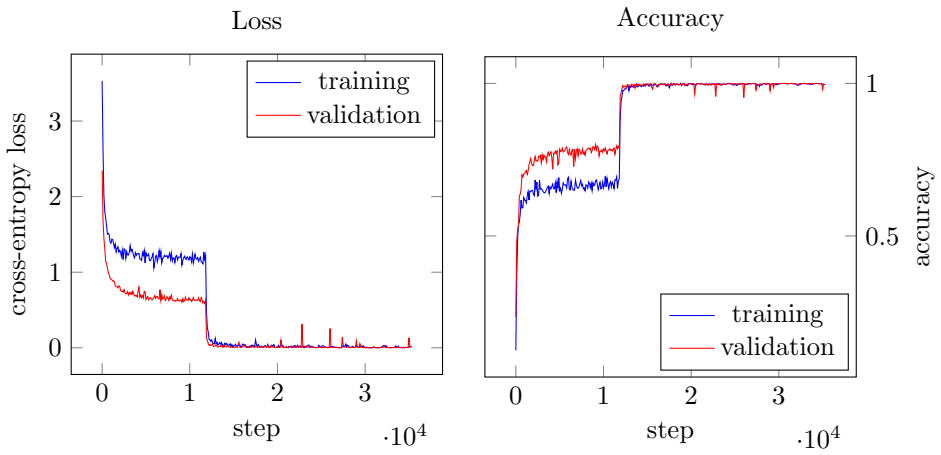


Figure 3.13: Loss and accuracy during training on both the training and validation sets for the InceptionV3 model. The best validation accuracy is 100.00%.

model	parameters	accuracy	errors	training accuracy
InceptionV3 [45]	$2.44 \cdot 10^7$	100.00%	0	99.98%
VGG [40]	$1.29 \cdot 10^8$	99.94%	2	99.84%
ResNet [42]	$1.12 \cdot 10^7$	99.91%	3	99.93%
AlexNet [33]	$5.71 \cdot 10^7$	99.02%	31	99.51%
CNN $(100, 3, 3, 2)$	$1.41 \cdot 10^7$	96.94%	97	99.62%
CNN $(100, 3, 3, 3)$	$1.44 \cdot 10^7$	96.90%	98	99.49%

Table 3.2: Performance of all piece classifiers on the validation set.

Chapter 4

Implementation

4.1 Data synthesis

4.2 Training

Chapter 5

Evaluation

5.1 Dataset

5.2 Critical appraisal

Chapter 6

Conclusion

6.1 Future work

Acronyms

ANN artificial neural network. 4

CAD computer-aided design. 4

CNN convolutional neural network. ii, 3, 4, 6, 12, 14, 15, 18

FEN Forsyth–Edwards Notation [3]. 5, 7, 8, 13

HOG histogram of oriented gradients. 2, 3

ReLU rectified linear unit. 14

RGB red, green, blue. 14

ROI region of interest. 17, 18

SIFT scale-invariant feature transform. 2

SVM support vector machine. 2

Bibliography

[1] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016.

[2] G. Kasparov and M. Greengard, *Deep Thinking: Where Machine Intelligence Ends and Human Creativity Begins*. 2018, ISBN: 978-1-4736-5351-1.

[3] S. J. Edwards, *PGN Standard*. Mar. 1994. [Online]. Available: `http://archive.org/details/pgn-standard-1994-03-12`.

[4] H. Baird and K. Thompson, 'Reading chess,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 6, pp. 552–559, Jun. 1990.

[5] I. M. Khater, A. S. Ghorab and I. A. Aljarrah, 'Chessboard recognition system using signature, principal component analysis and color information,' in *International Conference on Digital Information Processing and Communications*, Jul. 2012, pp. 141–145.

[6] A. Sameer, *Tensorflow_chessbot*, Sep. 2020. [Online]. Available: `https://github.com/Elucidation/tensorflow_chessbot`.

[7] A. Roy, *Chessputzer*, Jul. 2020. [Online]. Available: `https://github.com/metterklume/chessputzer`.

[8] V. Wang and R. Green, 'Chess move tracking using overhead RGB webcam,' in *International Conference on Image and Vision Computing New Zealand*, Nov. 2013, pp. 299–304.

[9] T. Cour, R. Lauranson and M. Vachette, 'Autonomous Chess-playing Robot,' École Polytechnique, Palaiseau, France, Jul. 2002. [Online]. Available: `http://www.timotheecour.com/papers/ChessAutonomousRobot.pdf`.

[10] D. Urting and Y. Berbers, 'MarineBlue: A low-cost chess robot,' in *International Conference Robotics and Applications*, Salzburg, Austria, Jun. 2003, pp. 76–81.

[11] N. Banerjee, D. Saha, A. Singh and G. Sanyal, 'A Simple Autonomous Chess Playing Robot for playing Chess against any opponent in Real Time,' in *International Conference on Computational Vision and Robotics*, vol. 58, Bhubaneshwar, India: Interscience Research Network, Aug. 2012, pp. 17–22.

[12] A. T.-Y. Chen and K. I.-K. Wang, 'Computer vision based chess playing capabilities for the Baxter humanoid robot,' in *International Conference on Control, Automation and Robotics*, Apr. 2016, pp. 11–14.

[13] J. Gonçalves, J. Lima and P. Leitão, 'Chess robot system : A multi-disciplinary experience in automation,' in *Spanish Portuguese Congress on Electrical Engineering*, 2005.

[14] R. A. M. Khan and R. Kesavan, 'Design and development of autonomous chess playing robot,' *International Journal of Innovative Science, Engineering & Technology*, vol. 1, no. 1, 2014.

[15] A. T.-Y. Chen and K. I.-K. Wang, 'Robust Computer Vision Chess Analysis and Interaction with a Humanoid Robot,' *Computers*, vol. 8, no. 1, p. 14, 1 Mar. 2019.

[16] C. Matuszek, B. Mayton, R. Aimi, M. P. Deisenroth, L. Bo, R. Chu, M. Kung, L. LeGrand, J. R. Smith and D. Fox, 'Gambit: An autonomous chess-playing robotic system,' in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 4291–4297.

[17] E. Sokic and M. Ahic-Djokic, 'Simple Computer Vision System for Chess Playing Robot Manipulator as a Project-based Learning Example,' in *IEEE International Symposium on Signal Processing and Information Technology*, Dec. 2008, pp. 75–79.

[18] J. Hack and P. Ramakrishnan, 'CVChess: Computer Vision Chess Analytics,' Stanford University, 2014. [Online]. Available: `https://web.stanford.edu/class/cs231a/prev_projects_2015/chess.pdf`.

[19] J. Ding. (2016). 'ChessVision : Chess Board and Piece Recognition,' [Online]. Available: `https://web.stanford.edu/class/cs231a/prev_projects_2016/CS_231A_Final_Report.pdf`.

[20] C. Danner and M. Kafafy. (2015). 'Visual Chess Recognition,' [Online]. Available: `https://web.stanford.edu/class/ee368/Project_Spring_1415/Reports/Danner_Kafafy.pdf`.

[21] Y. Xie, G. Tang and W. Hoff, 'Chess Piece Recognition Using Oriented Chamfer Matching with a Comparison to CNN,' in *IEEE Winter Conference on Applications of Computer Vision*, Mar. 2018, pp. 2001–2009.

[22] A. De la Escalera and J. M. Armingol, 'Automatic Chessboard Detection for Intrinsic and Extrinsic Camera Parameter Calibration,' *Sensors*, vol. 10, no. 3, pp. 2027–2044, 3 Mar. 2010.

[23] S. Bennett and J. Lasenby, 'ChESS – Quick and robust detection of chessboard features,' *Computer Vision and Image Understanding*, vol. 118, pp. 197–210, Jan. 2014.

[24] K. Tam, J. Lay and D. Levy, 'Automatic Grid Segmentation of Populated Chessboard Taken at a Lower Angle View,' in *Digital Image Computing: Techniques and Applications*, Dec. 2008, pp. 294–299.

[25] J. E. Neufeld and T. S. Hall, 'Probabilistic location of a populated chessboard using computer vision,' in *IEEE International Midwest Symposium on Circuits and Systems*, Aug. 2010, pp. 616–619.

[26] R. Kanchibail, S. Suryaprakash and S. Jagadish, 'Chess Board Recognition,' 2016. [Online]. Available: `http://vision.soic.indiana.edu/b657/sp2016/projects/rkanchib/paper.pdf`.

[27] Y. Xie, G. Tang and W. Hoff, 'Geometry-based populated chessboard recognition,' in *International Conference on Machine Vision*, vol. 10696, International Society for Optics and Photonics, Apr. 2018, p. 1 069 603.

[28] Y.-A. Wei, T.-W. Huang, H.-T. Chen and J. Liu, 'Chess recognition from a single depth image,' in *IEEE International Conference on Multimedia and Expo*, Jul. 2017, pp. 931–936.

[29] M. A. Czyzewski, A. Laskowski and S. Wasik. (Jun. 2020). 'Chessboard and chess piece recognition with the support of neural networks.' arXiv: 1708.03898.

[30] T. Romstad, M. Costalba and J. Kiiski, *Stockfish*, Sep. 2020. [Online]. Available: https://github.com/official-stockfish/Stockfish.

[31] M. Acher and F. Esnault. (Apr. 2016). 'Large-scale Analysis of Chess Games with Chess Engines: A Preliminary Report.' arXiv: 1607.04186.

[32] A. Mehta and H. Mehta, 'Augmented Reality Chess Analyzer (ARChess-Analyzer): In-Device Inference of Physical Chess Game Positions through Board Segmentation and Piece Recognition using Convolutional Neural Networks,' *Journal of Emerging Investigators*, Jul. 2020.

[33] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'ImageNet classification with deep convolutional neural networks,' *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[34] M. A. Czyzewski, A. Laskowski and S. Wasik, 'LATCHESS21: Dataset of damaged chessboard lattice points (chessboard features) used to train LAPS detector (grayscale/21x21px),' RepOD, 2018.

[35] J. Hou, 'Chessman Position Recognition Using Artificial Neural Networks.'

[36] M. Bilalić, R. Langner, M. Erb and W. Grodd, 'Mechanisms and neural basis of object and pattern recognition: A study with chess experts,' *Journal of Experimental Psychology*, vol. 139, no. 4, pp. 728–742, 2010.

[37] Q. Zhou. (May 2018). 'Pattern recognition in chess,' [Online]. Available: https://en.chessbase.com/post/pattern-recognition-in-chess.

[38] 64 Squares. (Feb. 2020). 'Magnus Carlsen Chess Games,' [Online]. Available: https://www.pgnmentor.com/players/Carlsen/.

[39] R. Szeliski, 'Image formation,' in *Computer Vision: Algorithms and Applications*, London: Springer, 2011, pp. 27–86, ISBN: 978-1-84882-935-0.

[40] K. Simonyan and A. Zisserman, 'Very Deep Convolutional Networks for Large-Scale Image Recognition,' in *International Conference on Learning Representations*, San Diego, USA, May 2015.

[41] D. P. Kingma and J. Ba. (Jan. 2017). 'Adam: A Method for Stochastic Optimization.' arXiv: 1412.6980.

[42] K. He, X. Zhang, S. Ren and J. Sun, 'Deep Residual Learning for Image Recognition,' in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp. 770–778.

[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, 'ImageNet: A large-scale hierarchical image database,' in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255.

[44] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge: Cambridge University Press, 2004.

[45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, 'Rethinking the Inception Architecture for Computer Vision,' in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp. 2818–2826.

Appendix A

User manual

```
cd /Applications/Blender.app/Contents/Resources/2.90/python/bin
./python3.7m -m ensurepip
./python3.7m -m pip install --upgrade pip
./python3.7m -m pip install python-chess
```

Appendix B

Ethics self-assessment form

There are no ethical issues raised by this project. The self-assessment form is attached on the next page.

UNIVERSITY OF ST ANDREWS
TEACHING AND RESEARCH ETHICS COMMITTEE (UTREC)
SCHOOL OF COMPUTER SCIENCE
PRELIMINARY ETHICS SELF-ASSESSMENT FORM

This Preliminary Ethics Self-Assessment Form is to be conducted by the researcher, and completed in conjunction with the Guidelines for Ethical Research Practice. All staff and students of the School of Computer Science must complete it prior to commencing research.

This Form will act as a formal record of your ethical considerations.
Tick one box

☐ **Staff Project**
☐ **Postgraduate Project**
☒ **Undergraduate Project**

Title of project

Identifying chess positions using machine learning

Name of researcher(s)

Georg Wölflein

Name of supervisor (for student research)

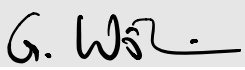
Dr Oggie Arandjelović

OVERALL ASSESSMENT (to be signed after questions, overleaf, have been completed)

Self audit has been conducted **YES** ☒ **NO** ☐

There are no ethical issues raised by this project

Signature Student or Researcher



Print Name

Georg Wölflein

Date

11.09.2020

Signature Lead Researcher or Supervisor

Print Name

Ognjen Arandjelovic

Date

15/09/2020

This form must be date stamped and held in the files of the Lead Researcher or Supervisor. If fieldwork is required, a copy must also be lodged with appropriate Risk Assessment forms. The School Ethics Committee will be responsible for monitoring assessments.

Computer Science Preliminary Ethics Self-Assessment Form

Research with human subjects

Does your research involve human subjects or have potential adverse consequences for human welfare and wellbeing?

YES ☐ **NO** ☒

If YES, full ethics review required
For example:
Will you be surveying, observing or interviewing human subjects?
Will you be analysing secondary data that could significantly affect human subjects?
Does your research have the potential to have a significant negative effect on people in the study area?

Potential physical or psychological harm, discomfort or stress

Are there any foreseeable risks to the researcher, or to any participants in this research?

YES ☐ **NO** ☒

If YES, full ethics review required
For example:
Is there any potential that there could be physical harm for anyone involved in the research?
Is there any potential for psychological harm, discomfort or stress for anyone involved in the research?

Conflicts of interest

Do any conflicts of interest arise?

YES ☐ **NO** ☒

If YES, full ethics review required
For example:
Might research objectivity be compromised by sponsorship?
Might any issues of intellectual property or roles in research be raised?

Funding

Is your research funded externally?

YES ☐ **NO** ☒

If YES, does the funder appear on the 'currently automatically approved' list on the UTREC website?

YES ☐ **NO** ☐

If NO, you will need to submit a Funding Approval Application as per instructions on the UTREC website.

Research with animals

Does your research involve the use of living animals?

YES ☐ **NO** ☒

If YES, your proposal must be referred to the University's Animal Welfare and Ethics Committee (AWEC)

University Teaching and Research Ethics Committee (UTREC) pages
http://www.st-andrews.ac.uk/utrec/