A Review of Computer Vision and Machine Learning Techniques to Digitise a Chess Position

Samuel Rapier
Student Number: 680029868

November 15, 2021

Signed: ______________________________
I certify that all material in this dissertation which is not my own work has been identified.

Abstract

Chess is currently rising in popularity online and thus also in over the board games. Many users have become used to analysing their chess games using online chess engines, however, struggle to do so for their in person games. We investigate techniques used to convert a real chess position into a digital equivalent using computer vision and machine learning. We compare corner-based and line-based board detection methods that have previously been implemented, finding that line-based approaches are preferred. We analyse previous methods of piece recognition using shape-based recognition, feature descriptor recognition and Convolutional Neural Networks, with CNNs being the best current solution. We also take a look at recent research around few-shot learning and how we could use this to classify chess pieces requiring smaller datasets that conventional CNNs.

1 Introduction

Chess has recently enjoyed a boom in popularity as seen by the increase in number of chess.com and lichess.org users as stated by their respective blog posts [17, 18]. In early 2020, lichess.org had peaks of 40,000 players, by December of 2020 this had risen to consistent peaks of 110,000 players. This is likely due to a combination of Covid-19 lock-downs, The Queen's Gambit on Netflix and many large content creators on YouTube and Twitch playing the game.

The rise of online chess will likely result in an increase in the number of over the board (OTB) chess players (those who play chess in person), especially as Covid-19 restrictions are lifted. The ability to recognise an OTB chess position and convert it to an online chess position would be very helpful to players. This would provide both the ability to use a chess engine and to record chess games. Chess puzzles are a common way for players to practice and some players prefer to practice these on a real board. However, this limits the players ability to analyse positions using a chess engine without having to spend time copying out the position.

Chess engines are used by players to analyse chess positions, indicating how to proceed and evaluating how strong a given position is. They can only be used on a computer since they perform large number of computations to evaluate a position and so require a position to be entered onto the computer.

Recording chess games as they progress is a task normally done by writing each chess move by hand until the game concludes. In high level games, they may make use of electronic chess boards, also known as DGT boards, which detects the pieces and automatically records the moves played. These can be very expensive and thus not realistic for amateur players. Recording the game by photo or video and automatically converting the position to a digital position would be a much cheaper solution in these cases.

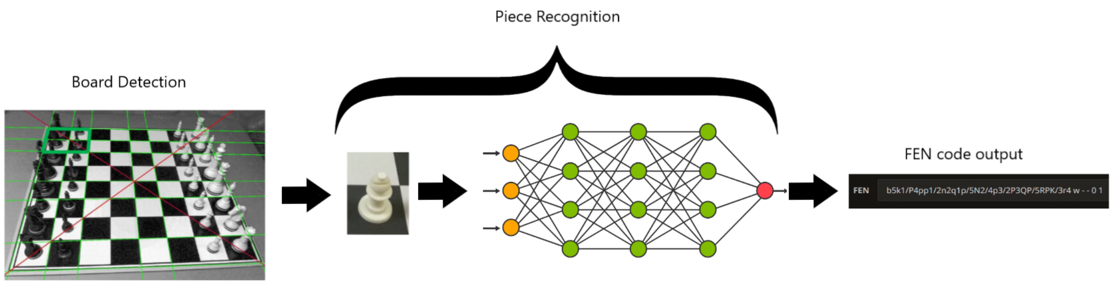
The proposed project will be able to convert a chess position in an image to the standard notation used for describing a chess position. This is the Forsyth–Edwards Notation (FEN) [19] and provides all the information necessary to start a game from a given position. FEN codes can be used by digital boards to rebuild the position. They can also be read by chess engines, such as Stockfish [16], to give an evaluation of a position.

Identifying chess pieces and chess positions remains a complex computer vision problem. It can be split into two main problems: identifying the chess board in an image and classifying the chess pieces on the board. The two main techniques of detecting a chess board are using corner-detection [1, 2, 6] algorithms and line-detection algorithms [3–6]. We shall analyse approaches previous attempts at chess board detection and determine the best approach to take. Previous approaches to classifying chess pieces include using shape-descriptors to identify a piece by its unique shape [4, 14], using feature descriptors [8], and using Convolutional Neural Networks (CNN) [10]. We shall analyse each of these approaches and decide which is the best current approach.

We will also take a look into recent advances in few-shot learning [21, 23] to see how well these can be applied to our project in the case of piece classification. This will help to understand how well the current approaches in few-shot learning can be implemented and how they manage detailed shapes such as chess pieces. It could help future research by uncovering some current limitations of few-shot learning and could also help to identify strengths of the approaches. It will be interesting to see how well a few-shot learning approach to classification using only a limited number of training images will compare to a CNN approach using a large dataset.

Literature Review

The problem can be broken down into smaller problems as outlined in Figure 1. Identify the board in the image, classify each of the individual chess pieces on the board, convert the information into an FEN code. We will identify the strengths and weaknesses of each technique and any interesting improvements previous authors have found to improve them. We shall also take a look at few-shot learning approach to classifying the pieces, which would be beneficial in the case where there is limited training data.



```
b5k1/P4pp1/2n2q1p/5N2/4p3/2P3QP/5RPK/3r4 w - - 0 1
```

Figure 1: Image Processing Pipeline to Digitise a Chess Position

2 Board detection using computer vision

Identifying the location of a chess board within an image is the first step towards understanding the chess position. This is a complex computer vision problem which can be approached using either corner-detection or line-detection.

A chess board is composed of 64 squares in an eight-by-eight grid formation with alternating coloured squares. Thus it consists of 9 horizontal lines and 9 vertical lines which can be used as domain-knowledge in both corner and edge based detection approaches. At the start of a game each player controls 16 pieces: one king, one queen, two rooks, two bishops, two knights and eight pawns.

Once the squares of the board are detected, we can crop the squares individually to include both the empty squares and those with pieces on. This would allow us to locate the position of the pieces once we classify them.

We must keep in mind that identifying the chessboard with pieces on the board can add extra complication since the chess pieces obscure vision of the board itself and potentially may obscure features of the board which are needed to confidently identify it.

Most approaches make use of the Hough Transform [15] which are able to detect straight lines in images even when there is missing data or noise in the image. The Hough Transform for line detection works out which of the possible lines that could pass through a point are most likely given the other features in the image.

2.1 Corner-based board detection

Corner detection approaches first detect the corners of the chess board itself and often perform the Hough Transform to identify the lines on the chess board. They proceed to locate the coordinates of the corners of each square on the chess board. This requires the chess board to be on a plain background to ensure that the lines detected are only those belonging to the chess board. Additionally, this requires the image to either be taken from a top-down view [2] or the board be empty of chess pieces to ensure that none of the corners are obstructed from view [1]. This approach works well for a game tracking solution, however, makes it difficult to take a snapshot of a position and determine the position, without first initialising the board.

Typical corner detection methods include SUSAN corner detector [27], Harris corner detector [28] and template matching-based corner detector [29]. According to Tam et al. [6] corner based approaches have a high tolerance against camera distortion since corners are less effected by warping. However, they are limited when corners are obscured and can be tricked by jagged lines caused by digitisation artifacts.

Koray [2] found that out of three test games, the corner points of the chessboard were successfully located in every game. Hack [1] found that in cases with appropriate lighting, the chessboard could be successfully located in 21 out of 25 test cases. However, this is greatly reduced in poor lighting conditions to 7 out of 15 test cases. Tam [6] found that using Harris-corner detection produced a false-positive rate of on average 85% since they were testing with pieces on the board, something neither Koray or Hack tested. This identifies to us that a purely corner-based detection method would not be suitable for our project.

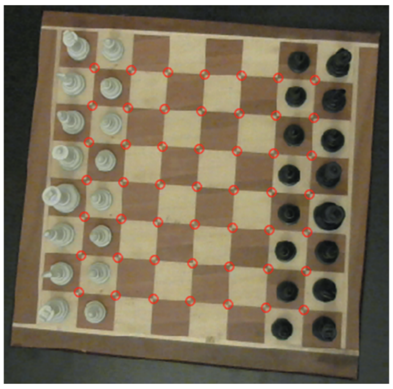


Figure 2: Corner detection example [2]

2.2 Line-based board detection

Common line-based approaches begin with edge detection. Edge detection is performed by detecting significant changes in pixel brightness in grey-scale images as explained by D. Ziou et al. [3]. Some examples of these include Canny edge detection [30] and Marr-Hildreth detectors [31]. Hough Transforms can then be used to identify the lines of the chessboard. Given that a chess board can be identified by 18 total lines and the orientation of half the lines will be perpendicular to the others makes them a popular approach [4,5]. They are also deemed a more robust approach since the lines are less likely to be completely obscured and it is more resistant to noise since a line will have significantly higher number of pixels than an individual corner.

Tam et al. [6] used a line based approach, using Canny edge detection and Hough Transforms to detect lines. They proceeded to use the geometry of a single square to extrapolate the locations of points of other squares. Their pure line-based approach produced decent results, with a false-positive rate around 15%, but was dependant on the angle at which images were taken. However, their results improved to a 100% hit rate, 0% miss and 0% false-positive, for all angles tested, once they extended the line based approach to find other squares.

A. De la Escalera [5] combined both a line-based approach and a corner-based approach to achieve good results at detecting the chessboard. However, as C. Danner et al. [4] identified, many irrelevant corners are detected and when chess pieces are on the board, many corners could not be detected, further reducing the usefulness of the corner-based approach. They had a success rate of 90% when taking photos from a steep angle, however, this reduced exponentially once the angle started to lower.

2.3 Evaluation of approaches

Overall, it would seem that the most accurate and robust approach to detecting the chessboard, with pieces obscuring squares and corners, would be a line-based approach. More specifically, Tam et al. [6] seemed to have a very strong solution with an improvement on a pure line-based approach that was not heavily dependant on the angle at which the chessboard is captured.

3 Piece Recognition

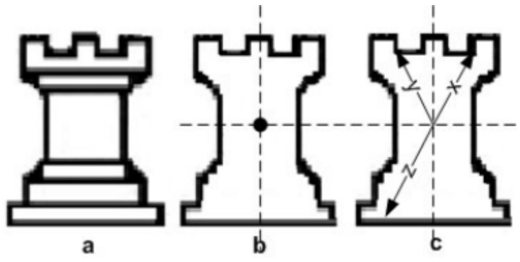
There are a number of game tracking applications which assume the starting positions of the chess pieces and detect the piece movements to build up the whole game. However, as previously stated, this would not work in our approach since we would like to be able to determine an unknown position from an individual image, without requiring the whole game to be played. Additionally, this approach would likely not work in other variations of chess, such as Fischer-Random [7] chess since the starting point of each of the pieces is randomised.

Therefore, we need an approach which will be able to classify each of the pieces on the board. Some techniques start by detecting the colour of the piece, and then use shape-descriptors to identify them fully [4,14]. A few more recent approaches have used machine learning classifiers to determine identify the pieces [8,10]. We will also look into using a few-shot learning techniques used with Convolutional Neural Networks which requires a very minimal dataset to learn from.

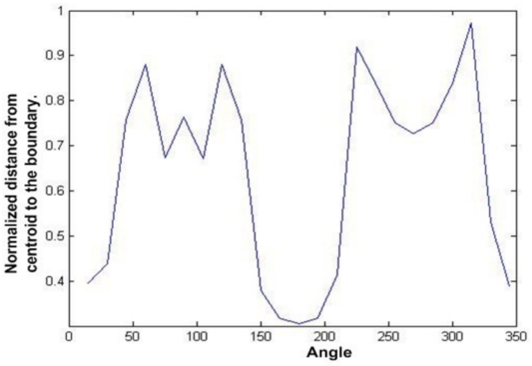
3.1 Shape-descriptor identifiers

Aljarrah et al. [14] focuses on identifying digital chess boards, however, their approach is still useful to analyse. They use a centroid function [33] to determine the signature of each shape. This was done by plotting the distance from the centroid to the shape boundary as shown in Figure 3a. Aljarrah used 23 points on each piece to calculate the signature, see Figure 3b, with which they then use to classify the pieces.

Aljarrah et al. had a 96.3% correct classification of pieces out of 107 pieces tested. Out of the four errors, only one piece was classified as the wrong type and the other three errors where from incorrect colour identification.



(a) samples of the signature values at three different angels [14]



(b) Signature of a rook [14]

Figure 3

C. Danner et al. attempts to recognise chess pieces on a real board and also uses shape recognition to identify the different pieces [4]. They determined that polygon approximation would not be the best approach since the difference in chess pieces comes in small changes in the shapes' boundaries, which does not work well since polygon approximation captures overall shape and ignores fine detail. Instead, they used Fourier descriptors for the pattern recognition to convert a 2-D object into a 1-D function which focuses on the shape contours. Danner uses the cumulative angular function [33] as the shape signature instead of the centroid distance function [33] that Aljarrah et al. [14] used since chess pieces have a similar base shape which makes centroid distance less accurate. This was not a problem that Aljarrah faced since the shape of pieces on a digital board are much more varied. The cumulative angular function instead focuses on the contours of the shape and so finds the finer details of the chess pieces.

Danner et al. found that their piece recognition approach was limited by the angle at which the pieces were captured and it would confuse rooks, bishops and kings when the viewing angle was too steep. However, using the correct viewing angle, all 42 pieces tested were correctly identified except for the king which had issues being identified when rotated. Danner explained that this was likely due to the cross on top of the king which caused a high discontinuity in the contours that were being used for identification.

3.2 Feature Descriptor approach

J. Ding [8] trained a classifier with gradient based feature descriptors. They experimented with both scale-invariant feature transform (SIFT) [25] and histogram of oriented gradients (HOG) [26]. Their implementation of SIFT involved using K-means clustering [34] over keypoints extracted by SIFT, they mapped these keypoints to the closest centroid. They used a histogram over the mapped keypoints and normalised it. This was done so the output matched that of the HOG implementation. They determined that HOG performed better than SIFT since HOG is well-suited to detection problems.

To classify the pieces, they used the feature descriptors from HOG as inputs into a modified sliding window technique [35]. Their implementation produced piece detection accuracy of 95% and piece classification accuracy of 85%.

3.3 Convolutional Neural Network approach

Quintana et al. [10] implemented a Convolutional Neural Network to classify chess pieces on a low-cost and low-power Nvidia Jetson Nano embedded device [32]. They used a high-level Keras API [12] on top of TensorFlow [11] to build and train the CNNs. They also implemented domain-knowledge to improve piece classification; taking into account all of the squares and pieces such that there could only be one king of each colour, bishops of the same colour should most often be on opposite coloured squares and other such chess rules. Quintana tested a number of different CNN models both with and without domain-knowledge included. They found that SqueezeNet-v1.1 and MobileNetV2(α= 0.5) were the most suitable models since they had an accuracy of over 90% whilst still maintaining an execution time under 1 second. The NASNetMobile and Xception models both had a higher accuracies however also had higher execution times, 3s and 5s respectively. On average the inclusion of domain-knowledge increased accuracy of classification by 1 to 4%. However, there were certain corner cases where including this knowledge decreased the accuracy. They gave the example of where the classifier would give two-squares a very high probability of containing a black-king, using domain-knowledge would select the square with the higher probability even though the black-king may be located on the square with lower probability. However, without the inclusion of domain-knowledge, the system would still give an incorrect result since it would assign a black-king to two squares.

3.3.1 What is a Convolutional Neural Network?

A Convolutional Neural Network (CNN) is an algorithm inspired by the visual cortex of the brain, these are some of the best algorithms to classify images [9]. CNN architectures generally consist of convolutional and pooling layers grouped together. These are followed by one or more fully connected layers. Figure 4 visualises the CNN architecture. If you input an image into the network, it enters the convolution and pooling stage. This produces a representation of the image which is then fed into one or more fully connected layers. The last of these layers outputs which class the input image belongs to.

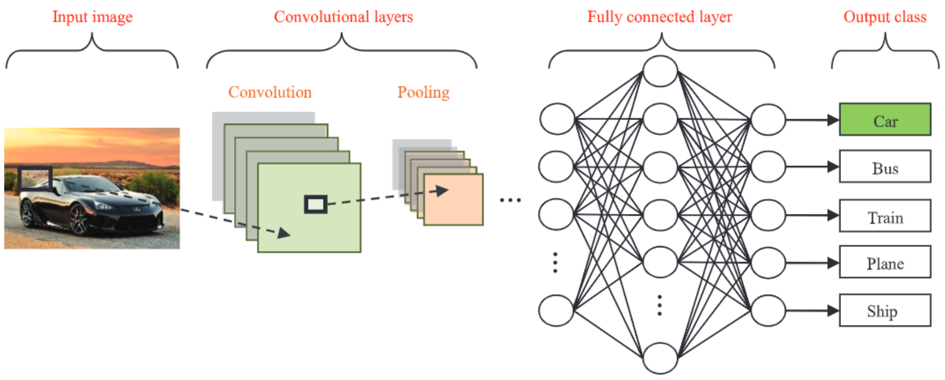


Figure 4: CNN image classification pipeline [9]

The convolutional layer analyses the influence of nearby pixels using a filter. Usually the filter is of size 3x3 or 5x5, it moves across the image from top left to bottom right. For each pixel in the input image, a value is calculated based on the filter using a convolution operation, Figure 5 demonstrates this. The collection of values calculated form a feature map. The feature maps are then used in an activation function to determine whether a feature is in the input image. Pooling layers can also be used to select the largest values in the feature maps to then be used as inputs into subsequent layers. Max pooling is often used as the operation in the pooling layer.

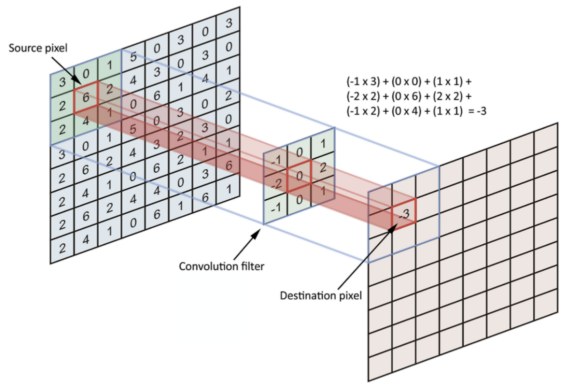

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
Destination pixel

Figure 5: Convolution filter example [20]

The Fully-Connected layer is where the classification is performed. The final feature maps from the convolutional layer are flattened into a column vector. This column vector is then used in a feed-forward neural network to classify the images.

3.4 Evaluation of approaches

The shape-descriptor approach seemed to work well except for when pieces were rotated which is a rather large limitation when it comes to classifying a whole chess board. In order to combat this, chess pieces would all have to be in the correct rotation before the image is captured which is not ideal. The feature-descriptor approach had a good detection accuracy but fell down when classifying the pieces with a classification accuracy of only 85%. Since there are 32 pieces on the board at the start of the game, an accruacy of 85% would mean there would likely be a miss classification in every three or four total board captures. The CNN approach had the highest piece classification accuracy at over 90% and was also well optimised. Additionally, the domain-knowledge added increased accuracy and robustness to the system.

4 Review of Few-shot learning

Convolutional Neural Networks are usually trained using a large labeled dataset which can be slow since they require many weight updates using stochastic gradient descent. Large sets of training data are not always available, this is where few-shot learning approach could be of interest. Few-shot learning (meta-learning, k-shot learning and one-shot learning all relate to the same topic) aims to generalise the learning abilities of an algorithm such that given only a few training images of a new object, it would be able to successfully classify that object.

4.1 Memory-Augmented Neural Network

A. Santoro et al. [21] uses a Memory-Augmented Neural Network (MANN) in their attempt to quickly learn new data and use this to make accurate predictions given only a few samples. MANNs can be split into three main parts, a controller, external memory, and read-write heads. The controller could be a feed-forward network or long short-term memory (LSTM) which is used for predictions. The external memory module stores the image representation and class label information together. This can then later be retrieved to make predictions when a sample from an already-seen class is presented. The read and write heads retrieve image representations from memory and place them into memory to be used later. This memory is used by the controller as the input to a classifier. Santoro at al. [21] also outline that in meta-learning, parameters are chosen to reduce the expected learning cost across a distribution of datasets. In contrast, standard Neural Networks try to choose parameters to minimise a learning cost across a dataset.

For their experiments, they trained their MANN using one-hot vector representations of classes. They trained on 100,000 episodes with five randomly chosen classes with randomly chosen labels. In test episodes, the network did no further learning and predicted class labels for unseen classes from the Omniglot [36] dataset. After just the second presentation of a class within an episode, they achieved an accuracy of 82.8%. This increased to 94.9% after five instances and 98.1% by the tenth (Figure 6). They used human performance as a baseline and discovered that the performance of the MANN exceeded that of a human. The MANN displayed better than random guessing on the first instance in a class and employed a strategy of educated guessing. A similar strategy was reportedly used by the human participants.

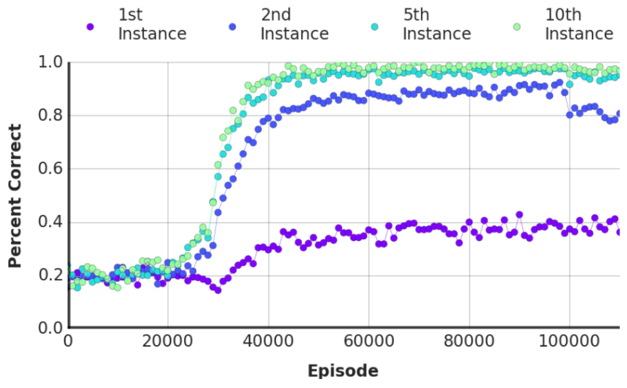


Figure 6: Omniglot classification using MANN, tested with five random classes per episode, one-hot vector labels [21]

4.2 Non-parametric one-shot learning

Vinyals et al. [22] demonstrates that the order of inputs and/or outputs has a significant effect when learning an underlying model. They discuss their developments of the sequence-to-sequence (seq2seq) framework [24] so that it can handle inputs sets properly. They proceed to propose a loss function that can deal with the lack of structure in the output sets by searching over possible orders during training.

In [23], Vinyals et al. continues his work to build a model that once trained could produce sensible labels for unobserved classes without any changes to the model. They took a non-parametric approach to solve one-shot learning. This took advantage of the set-to-set framework [22] to map a set of input-label pairs to a classifier which would define a probability distribution over outputs $\hat{y}$. They compute the probability over $\hat{y}$ by equation (1):

$$P(\hat{y}|\hat{x}, S) = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \tag{1}$$

$$a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x}), g(x_i))}}{\sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}} \tag{2}$$

where $\hat{x}$ is the test example, x_i and y_i are the inputs and the corresponding label distributions are from the set $S = (x_i, y_i)_{i=1}^{k}$. The attention mechanism is given by $a(.,.)$ which uses the softmax over the cosine distance c as defined by equation (2) where the embedding functions f and g are are neural networks, specifically deep convolutional networks for image tasks.

To train their model Vinyals used "episodes", each episode has a subset of labels L (e.g., L could be cats, dogs). They use L to sample a support set S and a batch B such that both S and B only contain labeled examples of cats and dogs. Their model is then trained to minimise the error predicting the labels in the batch B when only being given the support set S. This is classed as meta-learning since the training process learns to learn from a given support set to minimise the loss over a batch. They state that their model does not need fine tuning on classes it has never seen, however, when samples being tested diverge too far from the samples used to train the model, it will not work.

Vinyals tested their model using the Omniglot dataset in much the same way as Santoro. They used a simple CNN as the embedding function. They outperformed Santoro's MANN and produced a 98.1% accuracy for 1-shot and a 98.9% accuracy for 5-shot.

5 Conclusion

In summary, we determined that corner-based board detection is not suitable for our project since it is not accurate enough when chess pieces are on the board unless the photo is taken in a top-down configuration, this is because the pieces obscure the corners on the board. Line-based board detection methods are much more suitable for our project as the chess pieces do not cause so many issues. Tam et al. [6] had an intuitive improvement on a line-based approach to extrapolate all the squares on the board to improve accuracy and increase robustness. We discovered that Convolutional Neural Networks are currently the best approach to take to classify chess pieces. The shape-descriptor methods for piece identification do not work well enough when pieces are rotated and the feature descriptor approach is not as accurate as the CNN approach. We have also identified some approaches to few-shot learning which could be implemented into our project for piece classification. In scenarios where there is limited training data, few-shot learning could be a viable alternative since it does not require large datasets to learn from. Few-shot learning could be implemented into our system for piece recognition by training the model using batches of 5 chess pieces at a time. It will be useful to experiment with how well these techniques will work on chess pieces since they are fairly detailed in comparison to the objects that previous authors have tested so far.

Project Specification

6 Requirements

The aim of the project is to convert a chess position given in an image to an FEN code, the standard notation for describing a chess position. This is to be done using a combination of computer vision and machine learning techniques, with a key interest in exploring how well few-shot learning techniques could be applied to classify chess pieces. The requirements are as follows:

ID	**Requirements**
	Non-Functional Requirements
1	User should be able to upload an image using a mobile device
2	The system should display the digital chess position found, with the option for chess engine analysis
3	Images should be taken from predefined angles
	Functional Requirements
4	System should detect a chess board in a given image, provided one exists.
5	The system should resize and crop the image provided into individual images containing squares of the chess board.
6	The system should classify each of the individual pieces or empty squares correctly
7	The system should produce an FEN code from the classified pieces.

6.1 Evaluation Criteria

Requirement 1:
To test this the user should be able to upload an image stored on their mobile device as an input to the system. The mobile device should be either Android or iOS *(the project shall be limited to include only one of the two).* This will be deemed successful if the system can process the image.

Requirement 2:
Once a user has uploaded an image, the system should direct the user to lichess.org analysis page containing a valid chess position. Lichess already has the option to request computer analysis for a

given position. This criteria will be achieved if the user can see a digital chess board containing a chess position after they have uploaded an image and can successfully obtain an engine analysis of the position.

Requirement 3:
The angle at which the image is taken will be experimented with as part of the project. The image take should be within an optimal range that we discover through our project testing.

Requirement 4:
The success of this criteria is dependent on how accurately and consistently the chessboard can be recognised from the provided image. We are aiming for a board detection accuracy of 95% over 100 chess positions tested, each with between 10-32 pieces on the board.

Requirement 5:
We can measure the success of this requirement by analysing the cropped images. To test this, the cropped image will need to include the square and piece, without including other pieces in the image. The piece should be easily recognisable by a human. If 30 randomly selected images all pass this test, we will deem it a partial success. The system should also be recording the location of each of the squares as the image is cropped. This will be tested at the end when the final position is displayed. If 95% of pieces are on the correct corresponding squares on a digital board, irrespective of if the piece is classified correctly, this will be deemed a success.

Requirement 6:
The success of the classifiers should be measured by the classification accuracy against a set of 200 labeled test images, containing all piece types of each colour on different coloured squares, including empty squares. An accuracy of 90% will be considered a success.

Requirement 7:
If a digital chess board can read the FEN code correctly, this will be deemed a success. We shall test this using lichess.org since you can generate an analysis board on lichess.org by importing an FEN code.

7 Deliverables

We shall produce a chess board detection algorithm influenced by the implementation in [6], which should be robust enough to handle cases where features of the board used for identification are obscured. We will evaluate how well a conventional CNN will compare to a few-shot learning approach to classifying images of individual chess pieces. Making use of [10] to help design and build our conventional CNN and we will take inspiration from both [21, 23] for our few-shot learning approach. We will need to evaluate what current dataset exist that can be used to train a CNN for chess piece image classification and whether these could be suitable for few-shot learning. In addition, we shall experiment with the angle at which the chess board image is captured to find the most optimal position for both board detection and image classification combined.

References

[1] J. Hack and P. Ramakrishnan. (2014). "CVChess: Computer Vision Chess Analytics"

[2] C. Koray and E. Sumer, "A Computer Vision System for Chess Game Tracking," presented at the 21st Computer Vision Winter Workshop, Rimske Toplice, Slovenia, 2016

[3] D. Ziou and S. Tabbone, "Edge Detection Techniques - An Overview" 1998

[4] C. Danner and M. Kafafy. (2015). "Visual Chess Recognition"

[5] A. De la Escalera and J. M. Armingol, "Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration," Sensors, vol. 10, no. 3, pp. 20272044, 2010

[6] K. Y. Tam, J. A. Lay, and D. Levy, "Automatic grid segmentation of populated chessboard taken at a lower angle view," in Digital Image Computing: Techniques and Applications (DICTA), 2008. IEEE, 2008, pp. 294–299.

[7] https://en.wikipedia.org/wiki/Fischer_random_chess

[8] Ding, J. ChessVision: Chess board and piece recognition.

[9] Rawat, W., and Wang, Z.Deep convolutional neural networks for image classification: A comprehensive review.Neural Computation 29, 9 (2017), 2352–2449.

[10] Mallasén Quintana, D., Del Barrio, A. and Prieto Matias, M. (2020). LiveChess2FEN: a Framework for Classifying Chess Pieces based on CNNs.

[11] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro,C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefow-icz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga,R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J.,Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke,V., Vasudevan, V., Viegas, F., Vinyal, O., Warden, P., Watten-berg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scalemachine learning on heterogeneous systems, 2015

[12] Chollet, F., et al.Keras.https://keras.io, 2015.

[13] Bianco, S., Cad'ene, R., Celona, L., and Napoletano, P.Benchmarkanalysis of representative deep neural network architectures.IEEE Access 6(2018), 64270–64277.

[14] I. A. Aljarrah, A. S. Ghorab, and I. M. Khater, "Object Recognition System using Template Matching Based Signature and Principal Component Analysis", International Journal of Digital Information and Wireless Communications vol. 2, no. 2, pp. 156-163. The Society of Digital Information and Wireless Communications, 2012

[15] L. Chandrasekar and G. Durga, "Implementation of Hough Transform for image processing applications," 2014 International Conference on Communication and Signal Processing, 2014, pp. 843-847, doi: 10.1109/ICCSP.2014.6949962.

[16] https://en.wikipedia.org/wiki/Stockfish_(chess)

[17] https://lichess.org/blog/X-2TABUAANCqhnH5/lichess-end-of-year-update-2020

[18] https://www.chess.com/blog/erik/incredible-second-wave-of-interest-in-chess

[19] https://en.wikipedia.org/wiki/Forsyth%E2%80%93Edwards_Notation

[20] https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

[21] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks

[22] O Vinyals, S Bengio, and M Kudlur. Order matters: Sequence to sequence for sets.

[23] O Vinyals, C Blundell, T Lillicrap, K Kavukcuoglu, and D Wierstra. 2016. Matching networks for one shot learning. In Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)

[24] Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. Grammar as a foreign language. In Advances in Neural Information Processing Systems, 2015

[25] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International journal of computer vision, vol. 60, no. 2, pp. 91-110, Nov. 2004

[26] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2005

[27] S. M. Smith and J. M. Brady, "SUSAN - A New Approach to Low Level Image Processing", in International Journal of Computer Vision, 1995

[28] C. Harris and M. Stephens (1988). "A combined corner and edge detector", Proceedings of the 4th Alvey Vision Conference

[29] R. Brunelli, Template Matching Techniques in Computer Vision: Theory and Practice, Wiley

[30] J. Canny, "A Computational Approach to Edge Detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986

[31] Marr, D.; Hildreth, E. (29 Feb 1980), "Theory of Edge Detection", Proceedings of the Royal Society of London. Series B, Biological Sciences

[32] https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/

[33] Zhang, D., & Lu, G. (2002). A Comparative Study of Fourier Descriptors for Shape Representation and Retrieval.

[34] https://en.wikipedia.org/wiki/K-means_clustering

[35] Laanaya, Hicham & Martin, Arnaud & Aboutajdine, Driss & Khenchaf, Ali. (2008). Classifier fusion for post-classification of textured images. Information Fusion - INFFUS. 1 - 7. 10.1109/ICIF.2008.4632257.

[36] BM Lake, R Salakhutdinov, J Gross, and J Tenenbaum. One shot learning of simple visual concepts. InCogSci, 2011