Determining Chess Board State Using Computer Vision Techniques to Predict Next Best Move

Moath Rabeh Khaleel
Faculty of Engineering
University of Jordan
Amman, Jordan
Masters in AI and Robotics

Mohammad Jamal AlKhatib
Faculty of Engineering
University of Jordan
Amman, Jordan
Masters in AI and Robotics

***Abstract*— this paper aims to design an image classification system that can determine the state of a chess board. The state of a chess board can then be fed to a chess engine like Stockfish to determine the next-best-move for a player. Pretrained models will be used as input to a fully connected layer. The optimal architecture found for the system will be discussed in detail.**

I. INTRODUCTION

When Artificial Intelligence took the world by storm in the second half of the 20th century, chess was one of the very first applications artificial intelligence was employed in. Building a machine that can beat humans was a stroke of imagination. As machines became more powerful, more algorithms could be fed to computers. Soon, the first computer was able to defeat the world champion at the time; Garry Kasparov [1].

Chess engines have become very powerful and robust. Engines can now determine the next best move in a chess game better than humans. The only thing the chess engine needs is an input determining the board state. Feeding a chess engine with a board state is still mostly done by humans.

This paper aims to bridge the gap of needing a human feeder by employing computer vision techniques to predict the state of a board from an image. The system can serve as a chess trainer for players. The computer vision system can also be coupled with a mechanical system, enabling players to play with a robot.

There are three key subsystems that must be established to build a system that can predict the next-best-move in a chess game; enhance the clarity of the chess board image captured, , and determine the type of piece occupying the square – if any. Computer vision techniques are ideal to serve such purpose.

Computer vision has seen tremendous growth in recent years. It has become interconnected with our day-to-day lives. Computer vision has been making light-speed improvement in accuracy and precision. Object classification is a subfield of computer vision and is the perfect candidate for determining the kind of piece occupying a square. Pre-trained computer vision models can be used to determine the state of a chess board by simply providing real-time photos of the chess board.

The introduction of deep neural networks eliminated the need for feature extraction seen in classical machine learning techniques. Computer vision enables feature extraction optimization and object classification through an architecture of convolutional, pooling, and fully connected layers.

In a paper published by Wolflein et al. [2] Hough transform was used to determine the edges of a chess board. Once edges were determined, an algorithm was used to divide the chess board to 64 pieces. Computer vision was used to determine if a square is empty, and if not, classify the piece occupying that space. Authors were able to achieve remarkable accuracies by using a pretrained Inception V3 model. The system was able to achieve an end-to-end accuracy of 99.7%.

In a paper published by Chen et al. [3], 500 images were used to test an object detection system for chess pieces. The paper discusses using Canny Edge Detection and morphological closing to determine the edges of the board and eliminate any shadows. The authors were able to achieve a per square accuracy of 92.2 %. The approach used by the authors was based on differential comparison between the previous and current state board. This was then used to feed a robot the next best move.

The computation of the next best move can be done using Stockfish. Stockfish is a very popular chess engine. Stockfish is an open-source engine that has been trained over millions of chess games. The engine can provide the next-best-move given a chess board state. Stockfish is a 13 times winner of the top chess engine championship [4].

This paper focuses on building a model that can predict the type of chess piece occupying a square. Nevertheless, the use of canny edge detection and Hough transform to segment the board into 64 squares is discussed in brief. The dataset used for the model assumes the squares are segmented already, as the main purpose was to test the piece classification prediction accuracy. Pretrained models will be used and built with a fully connected layer for image classification.

II. APPROACH

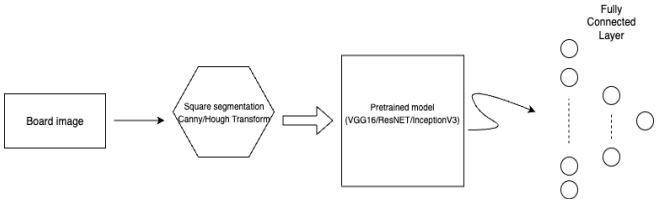


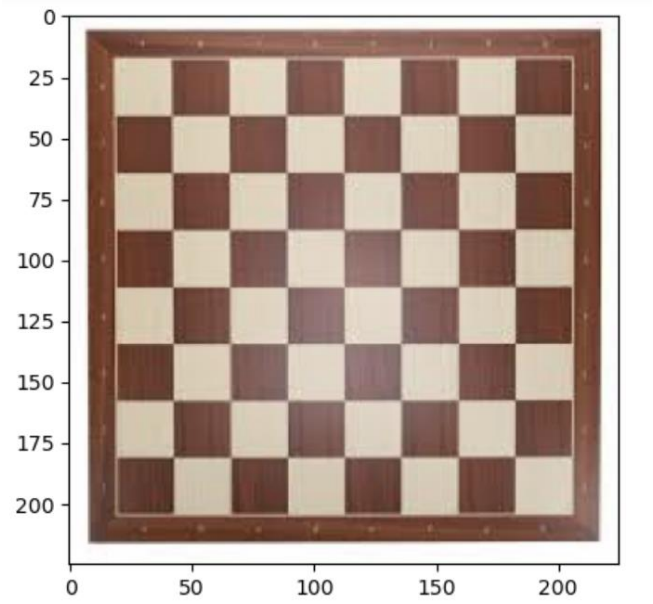
Figure 1 System Architecture

Given an image of a chess board, the image can be segmented into 64 squares using Canny edge detection and Hough Transform. Once edges and lines are determined, the points of intersection between lines can be considered for square segmentation. The 64 square images are fed to an image classifier to determine the piece occupying the square. This approach presents the advantage of knowing the location of each square, which helps determine the position of each piece – needed for board state.

Pretrained models are readily available and can be used for feature extraction. This paper tested popular models including VGG16, InceptionV3, and ResNET. Those models have already been trained on huge datasets and their architecture has been optimized. In this paper the models were used with and without training and results were compared.

The output of the model was fed to a fully connected layer. Different numbers of layers and neurons were used. Different overfitting reduction techniques were employing, including dropout, batch normalization, regularization, and data augmentation.

Adam and Stochastic Gradient Descent as compilers. Different momentums and learning rates were used. Early stopping was also employed along other callbacks.




```
plt.imshow(new_img)
```

```
<matplotlib.image.AxesImage at 0x7fd9e183d340>
```

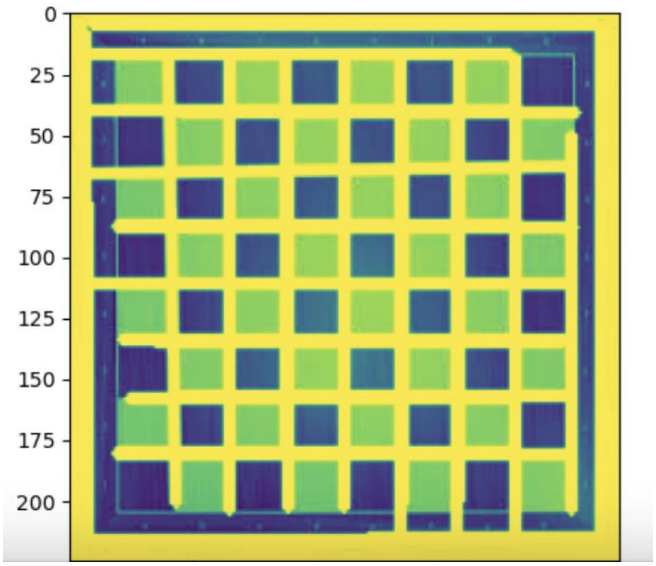


Figure 2 Hough Transform and Canny edge detection for square segmentation

Since this is an image classification problem, categorical cross-entropy was used as a loss function. Categorical accuracy was used as a metric to measure the accuracy of the model on the training, validation, and testing sets.

III. Data Description

A. *Chosen Dataset*

Different datasets are available online and can be used for different applications. Initially, the dataset in hand was found online and consisted of 2,406 images divided to train, validation, and test sets. After testing the dataset with the different models, it was determined that the dataset – along other online uploaded ones – is of low quality.

It was decided to build a new dataset by taking real-time photos of chess pieces. In total, 2,235 unique images were taken by our team. The images were divided into 13 distinct classes. The classes are black pawn, white pawn, black castle, white castle, black rock, white rock, black bishop, white bishop, black queen, white queen, black king, white king, and empty square. The new dataset gave better results as it had images of the pieces taken from different angles, with different levels of lighting, and different positioning on the chess board.

B. *Base Code Source*

For programmatic operations, Python 3.11 was used. Python is an object-oriented language that enables efficient and quick computation. Python also offers several libraries that can be used with machine learning.

Data enhancement was tested using Python's NumPy library. Python's Open CV library was used to build a classification model. Transfer learning was employed to use different pre-trained models, such as Inception V3 and VGG16.

Model training, compiling, and prediction was done with the help of the Tensor-flow library. Tensor-flow is a powerful library that allows building fully connected neural networks with ease.

IV. Experiments

As mentioned earlier, readily available datasets online were not performing well with the built architecture. In addition, most datasets were unrealistic and did not resemble real-life chess board photos. Hence it was decided to build a new dataset from scratch.

In total, 2,235 images were used for training, validation and testing. The images consisted of different pieces occupying a single chess board square. Images were taken from different angles and under different light intensities. The dataset resembles the segmented chess board obtained from Canny edge detection and Hough transform.

There are 13 distinct classes a square state can be in. The obvious choice for measuring the accuracy of the model was through categorical cross-entropy. The class an image belongs to was labeled using Tensor-flow's out of the box Image Data Generator. Therefore, the quantitative measurement of the model performance was possible.

Different fully connected models were used throughout experimentation. VGG16, InceptionV3, and Resnet were among the transfer models used. Having many layers with fewer neurons was compared with having fewer fully connected layers but with many neurons. Different learning rates and epoch sizes were tested. Different optimizers were used, such as Adam and Stochastic Gradient Descent.

Successive experiments were compared and results were built on.

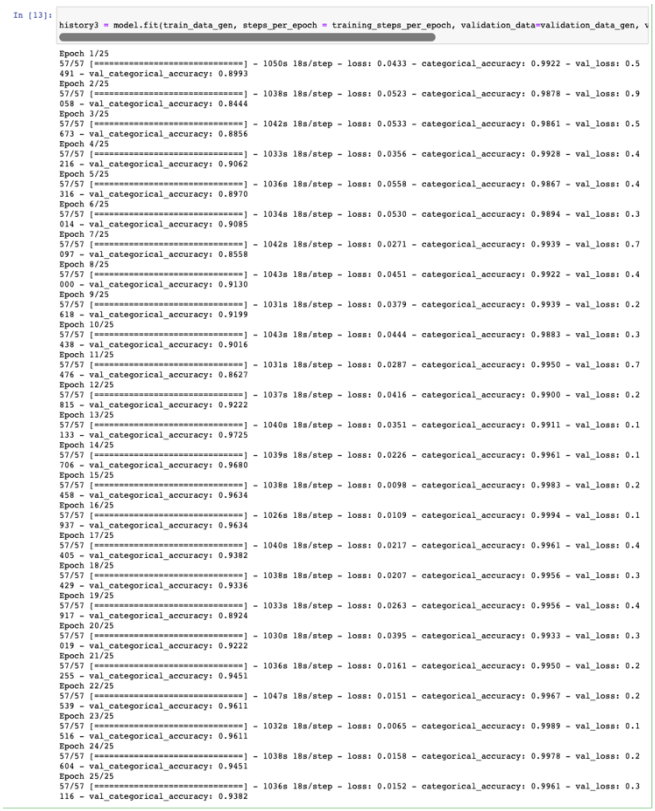


Figure 3 Best performing model results

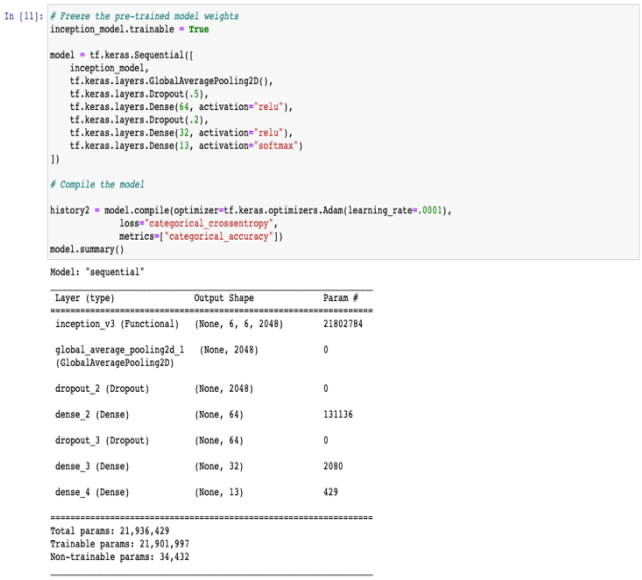


Figure 4 Best performing model summary

Initially, pre-trained models were used. The results were not satisfactory. Therefore, the VGG16 and InceptionV3 models were trained all over again. This enhanced performance significantly.

V. Implementation

The canny edge and Hough transform code used to test segmenting a chess board was done by our team. We did not write the algorithm that actually segments the board into squares, but it could be done as an enhancement in the future.

Creating the dataset, reading them to a Jupiter notebook, cleaning any corrupt files, and writing code for building the sequential model were all done by our team.

The VGG16, InceptionV3, and Resnet models were used as provided by Tensor-flow. The fully connected layer added afterward, as well as any extra pooling and convolution layers used, were added by our team.

VI. Conclusion

The model built performed relatively well in classifying the chess piece occupying a chess board square. The model performed better than most past literature found on the topic. Table I shows the best accuracies that we were able to achieve.

TABLE I. Comparison of Different Models

Model	Number of Trainable Parameters	Accuracy	Validation
Transfer learning , InceptionV3	21,936,429	0.9989	0.9725
Transfer learning , (More Layers and neuruns)	22,341,613	0.8326	0.9634
Transfer learning , VGG16	4,203,917	0.9390	0.8559
Transfer learning , VGG16(Sparse)	3,374,849	0.9772	0.7025

There is a lot of room for improvement in the future. Improvement is mainly related to the board segmentation step. In our current implementation, we did not include an algorithm to segment an image of a board into 64 squares. If done, the classification model can be tested. The model may perform differently depending on the quality of segmentation.

There is also room for improvement in the overall accuracy obtained. More layers can be added if more computational power becomes available at our disposal.

REFERENCES

[1] *In 1997, an IBM computer beat a Chess World Champion for the first time | CNN business* (2022) *CNN*. Available at: https://edition.cnn.com/videos/business/2022/05/11/ibm-deep-blue-computer-beats-garry-kasparov-chess-champion-1997-vault-jg-orig.cnn

[2] Wölflein, G. and Arandjelović, O. (2021) *Determining chess game state from an image*, *arXiv.org*. Available at: https://arxiv.org/abs/2104.14963

[3] Chen, A.T.-Y. and Wang, K.I.-K. (2019b) *Robust computer vision chess analysis and interaction with a humanoid robot – DOAJ*, *Computers*. Available at: https://doaj.org/article/958c862bf04f478582a9963c8a92ca03 (Accessed: 05 June 2023).

[4] Maharaj, S., Polson, N. and Turk, A. (2021) *Chess AI: Competing paradigms for machine intelligence*, *arXiv.org*. Available at: https://arxiv.org/abs/2109.11602.