# Chess Board and Piece Recognition

**Anurag Shenoy**
CISE
University of Florida
shenoy.a@ufl.edu

**Ravi Teja V**
CISE
University of Florida
rveernapu@ufl.edu

## Abstract

This paper explores Faster RCNN and Multiclass classification algorithms to detect and recognize chess pieces from a real life image of a chess board. The recording of moves during a chess game is a tedious manual task that impedes the flow of the game, especially in formats such as blitz chess where manually recording moves hinders efficient use of time. The automated detection and classification of chess pieces can help players to digitize or save their games and resume it when needed. Using the very deep VGG-16 model and ImageNet's initial weights, our detection system has an excellent mean Average Precision (mAP) of 97.47%.

## 1 Introduction

The recording of moves during a chess game is a tedious manual task that impedes the flow of the game, especially in formats such as blitz chess where manually recording moves hinders efficient use of time. At the professional level, specialized chess sets have been developed to record moves automatically. But this equipment is expensive and not easily accessible to recreational players. Further, in an age when much analysis and storage of chess games is done on computers, chess players at all levels can benefit from the ability to input games into a computer-readable format by simply taking a picture of a real-life board, as opposed to manual input. In recent times, chess-playing AI have relied on either specialized equipment or human assistance in order to function. A fully autonomous chess-playing robot would require the ability to detect a chessboard and pieces through vision.

In this paper, we discuss an approach that takes an image of a chessboard as input, and enables a computer representation of the chessboard as an output. This approach is best suited for recreational chess players. In such settings, the players do not have multiple camera angles or an overhead camera. So the option of having images at multiple angles was ruled out. This approach was devised with the assumption that the players takes a picture from a mobile devices, from one side of the board, i.e. from the players perspective or from the side.

In the remainder of this paper, we will outline the algorithm and methods used in this approach, discuss the experimental setup, results and conclusions.

## 2 Problem Statement

In this project, our objective is to explore and use a Convolutional Neural Network to extract features from a dataset of images consisting of chess boards, propose regions of interest and then classify the detected objects as belonging to the classes of chess pieces. We will primarily use the Mean Average Precision (mAP) score to determine how well the network is able to detect the various chess pieces in the images.

In this project we explore a way to build a model to faithfully reproduce a chess position when the model is given a real life chess image as an input. The realm of chess recognition can be broadly

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

separated into two major areas. Board recognition refers to the detection of the chess board within the image and the identification of board characteristics, such as the orientation, the location of squares, etc. This usually involves finding a projective transformation that rectifies the image into a format where the location of the board is known. Piece recognition refers to the detection of pieces on the board and the localization and classification of those pieces. Since piece recognition relies on knowledge about the board, board recognition is typically a prerequisite step to piece recognition.

## 3 Dataset

The dataset we used is the Roboflow Chess Pieces Recognition dataset available here: `https://public.roboflow.com/object-detection/chess-full`

All photos were captured from a constant angle, a tripod to the left of the board. The bounding boxes of all pieces are annotated as follows: *white-king*, *white-queen*, *white-bishop*, *white-knight*, *white-rook*, *white-pawn*, *black-king*, *black-queen*, *black-bishop*, *black-knight*, *black-rook*, *black-pawn*.

There are 2894 labels across 292 images for the raw version, and the updated version has 693 images.

Table 1: Dataset Train Test Split

| Train | Valid | Test |
|---|---|---|
| 606 | 58 | 29 |

Figure 1: Sample images present in the dataset

## 4 Algorithm

We are primarily using Neural Networks in this paper to perform the feature extraction from the images of Chess boards, detection of Chess Pieces on the board and then to classify each piece in order to enable the construction of a virtual chess game board.

### 4.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are similar to ordinary artificial neural networks, as they both are made up of neurons with trainable weights and biases. CNNs have three main layers, namely

Convolutional Layer, Pooling Layer, and Fully-Connected Layer. We have stacked these layers to form a large Convolutional Neural Network.

The Convolutional Layer convolves the image with a filter, producing multiple feature maps for each image. Once we obtain feature maps, we apply an activation function such as RELU on these features. The convolution layer's parameters consist of filters of a fixed size which are slid across the input image. At each convolution pass, we compute the dot product of the filter and the input image, which produces a 2-D activation map, which gives us the result of the dot product of that filter at each position.

**Convolution Layer:**

$$Y = W * X$$

$$Y_{ij} = \sum_k \sum_l W_{ij} X_{i-k,j-l}$$

Where $W$ is the window and $X$ is our input image.

**RELU:**

$$a(u) = max(0, u)$$

$$\dot{a}(u) = \begin{cases} 1, & u > 0 \\ 0, & u \leq 0 \end{cases}$$

**Optimization:**

$$\underset{W_1,\ldots,W_Q,\Theta}{\text{minimize}} \frac{1}{n} \sum_{i=1}^{n} l(y_i, \Theta^T a(W_Q a(\cdots a(W_1 x_i)))) + \lambda \sum_{q=1}^{Q} \|W_q\|^2 + \lambda \|\Theta\|^2$$

The second type of layer used is a pooling layer which reduces the dimensions of the feature maps by performing a down-sampling operation using a filter. Pooling not only reduces the dimensions of the feature maps, but also reduces the number of parameters in the network in order to control over-fitting.

Given an input of width $W_1$, height $H_1$ and depth $D_1$, a filter/kernel size $K = 2$, and a stride $S = 2$, max pooling will produce an output with dimensions:

$$W_2 = (W_1 - K)/S + 1$$

$$H2 = (H1 - K)/S + 1$$

$$D2 = D1$$

The third type of layer is a Fully Connected layer, where each neuron is connected to all the other neurons in the previous layer and this type of layer is used to compute the class scores.

### 4.2 Feature extraction

We are using a VGG-16 model for the purpose of feature extraction.

We load the ImageNet VGG16 trained model's weights into our VGG feature extraction network.

The first two blocks of two convolutional (conv.) layers followed by a pooling layer are kept frozen and are untrainable.

During training, we provide input images of size 416 x 416 with 3 channels one for each of RGB. The pre-processing we do is to scale the image so that the smaller dimension is 600 pixels and then we subtract the mean rgb value of the ImageNet dataset from each pixel.

The image is then passed through multiple blocks of conv. layers, where filters of size 3 x 3 are used. The pooling is done by 4 max-pooling layers each with a stride of 2 and a window of size 2 x 2.

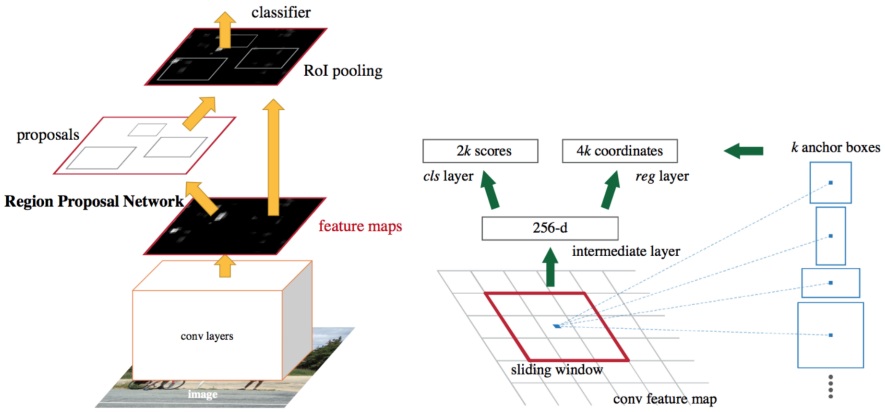


Figure 2: Faster RCNN Architecture and

### 4.3 Region Proposal

The region proposal network (RPN) generates a box at each anchor consisting of a score (the objectness score) and coordinates. The published implementation uses a binary classification scheme with two mutually exclusive one-hot encoded class outputs per anchor: object and background. Our implementation applies sigmoid activation to each pair of signals for every anchor. Each bounding box is labelled as belonging to either the background (0.0) class or the object class (1.0). If a box has a score of greater than 0.5, it is considered as an object. These boxes are then passed to a labelling stage where they are compared to ground truth object boxes to check whether they sufficiently overlap. If they do, then the box is labelled according to the class of the ground truth box, otherwise it is labelled as background. The labelled boxes called proposals, or regions of interest (roi) are then passed to the detector stage.

### 4.4 Classification

The Detector is responsible for Region of Interest (roi) pooling and then the final classification using fully connected layers. The region proposals are cropped to size 14 x 14 using bilinear interpolation, and then pooled using a max-pooling layer with a window of size 2 x 2 to reduce the size of the proposals to 7 x 7. These proposals are then passed through two full connected dense layers with optional dropout layers which are connected to a fully connected classifier layer with a softmax activation and a fully connected regressor with a linear activation for the bounding boxes.

Classification is performed by the fully connected classifier layer mentioned above, and we obtain the classes for each roi. The regressor determines the bounding boxes which have the highest classification and bounding box overlap scores.

### 4.5 Mean Average Precision

To determine if a prediction for a bounding box is correct (is similar to the ground truth box), we have to use a metric such as Intersection over Union (IoU). Intersection over Union is defined as

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

If the IoU is well above the threshold set (say 0.5), then it can be classified as a True Positive (TP), and if it is well below, then it is a False Positive (FP). We can thus calculate the precision and the recall, which we can use to calculate the Average Precision.

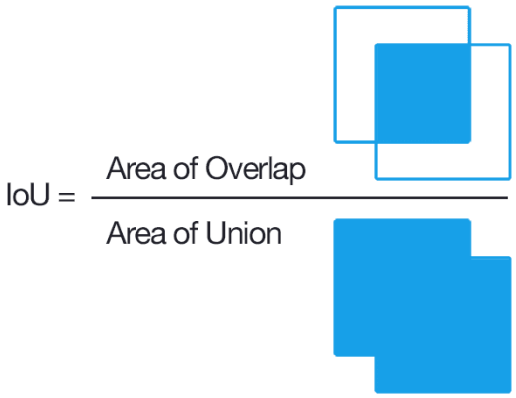


Figure 3: Intersection over Union

Average Precision (AP) is the area under the precision-recall curve.

The mean Average Precision or mAP score is calculated by taking the mean of AP over all the classes present in the dataset.

## 5 Experiments

The model training was performed with the following methods, each of which is explained further with the parameter information.

- SGD (Stochastic Gradient Descent)
- Adam Optimizer
- SGD with dropout

We trained the model using the following parameters:

Table 2: Model Parameters

| optimizer | learning rate | clipnorm | momentum | beta1 | beta2 | weight decay | dropout |
|---|---|---|---|---|---|---|---|
| adam | 0.001 | 0 | 0.9 | 0.9 | 0.999 | 0.0005 | 0.0 |

Initially we trained it for 20 epochs, which resulted in a mean Average Precision of 94.51%.

As both the train and test mAP scores were improving and the losses were decreasing, we trained it for a further 10 epochs for a total of 30 epochs which resulted in a mean Average Precision score of 97.47%.

A mAP score of 97.47% on the test set is surprisingly good and the model had low variance as it was able to achieve a mAP score of 95% on the train dataset as well.

**Optimizer Experiments**: Training the model using **adam** and **sgd** optimizers to see if we could improve the performance of the model.

SGD has recently received a large amount of attention for large neural networks and so we saw it fit to experiment with trying SGD optimizer in place of Adam optimizer.

Table 3: Optimizer Experiment Results

| Optimizer | RPN cls loss | Detector cls loss | Detector regr loss | Total Loss | mAP |
|---|---|---|---|---|---|
| adam | 0.0646 | 0.0657 | 0.0782 | 0.25 | 97.47% |
| sgd | 0.0215 | 0.0240 | 0.0184 | 0.09 | 97.30% |

We can see the training progress over 30 epochs in the following charts:

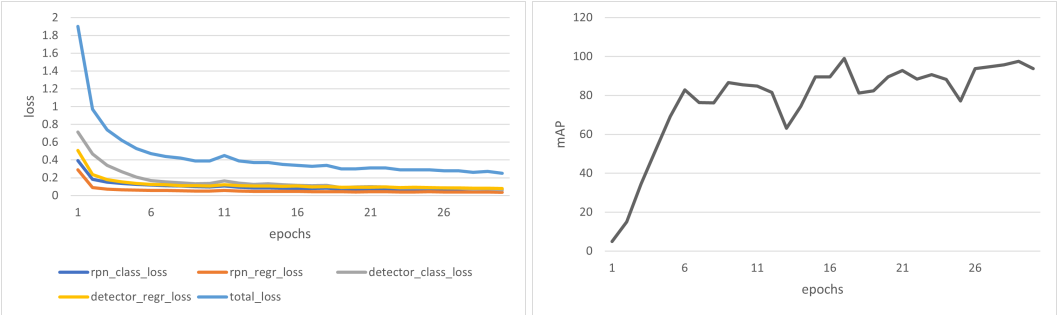


Figure 4: Training progress loss and mAP vs epochs (using Adam)

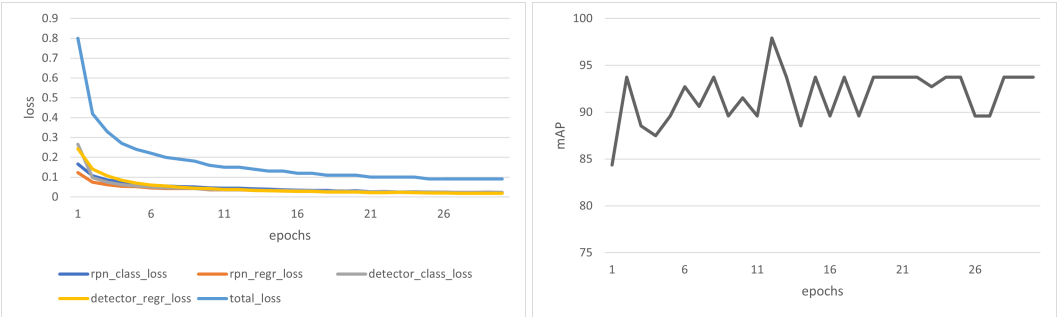


Figure 5: Training progress loss and mAP vs epochs (using SGD)

**Dropout Experiments**: As adding dropout allows us to randomly ignore a proportion of neurons when back-propagating. This also has the added effect of preventing overfitting by encouraging the network to actually learn a sparse representation as a side-effect.

We can see the effect of dropout on the training in the following charts.

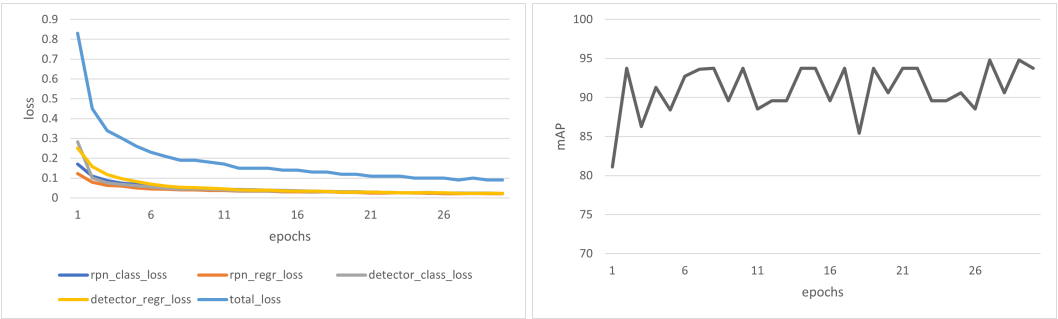


Figure 6: Training progress loss and mAP vs epochs (using SGD and 30% dropout)

## 6 Results

We were able to detect and classify chess pieces on a board in order to facilitate chess players in digitizing their game and saving the state of the game for future use.

The following table summarizes the test data precision for each class:

Table 4: Class-wise Precisions

| Class | Precision (%) |
|---|---|
| black-king | 100.0 |
| black-queen | 92.9 |
| black-rook | 99.0 |
| black-bishop | 95.9 |
| black-knight | 99.8 |
| black-pawn | 96.6 |
| white-king | 99.6 |
| white-queen | 98.6 |
| white-rook | 95.3 |
| white-bishop | 98.2 |
| white-knight | 99.7 |
| white-pawn | 94.1 |

While the model performs really well, it is not always perfect.

Figure 7: Model isn't perfect

In the above images we can see that the Black Rook in the upper image and the White Pawn in the lower image have been missed by the model.

The model misses a few pieces, but is overall very impressive as we achieved a mAP score of 97.47%.

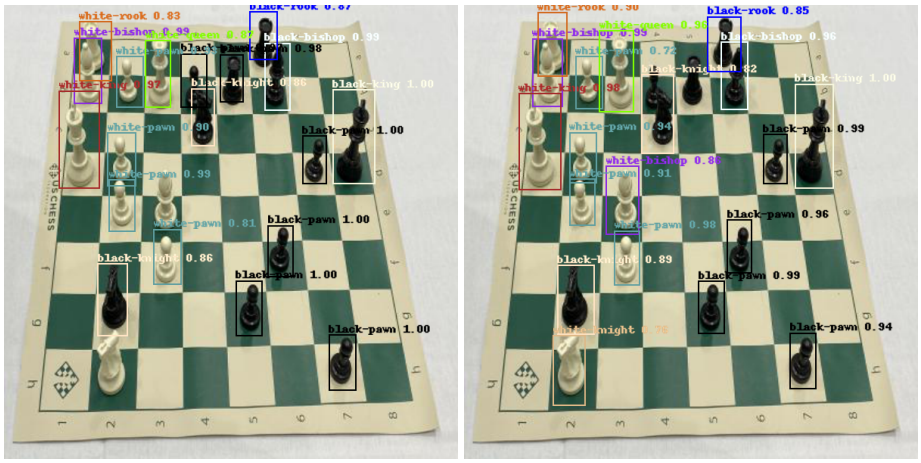

Figure 8: Detection of pieces in the test folder of the dataset

## 7 Conclusion

The source code for this project can be found at `https://github.com/shenoy-anurag/Chess-Vision`. In this paper, we have provided a proof of concept of using a CNN for chess board and piece recognition, which is more robust compared to traditional segmentation-based approaches which rely on heavy feature engineering. The training classifier produced an average precision of 97.47% on the test set. However, a few improvements have to be made to use this in real life scenarios. From the class-wise precision break down 4, we observe that pieces with rotational symmetry, such as pawns, rooks, queens and bishops had a lower precision. While adding more training data may improve the model performance, additional work in the feature extraction space may yield significant improvement in the model performance. This should be the direction for future work.

## References


[1] Ross Girshick, Jeff Donahue, Trevor Darrell & Jitendra Malik (2013) Rich feature hierarchies for accurate object detection and semantic segmentation

[2] Shaoqing Ren, Kaiming He, Ross Girshick & Jian Sun (2015) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

[3] Maciej A. Czyzewski, Artur Laskowski & Szymon Wasik (2017) Chessboard and chess piece recognition with the support of neural networks