Chess Pieces Detection using Detectron2

Sarath Kumar Mannam
College of Engineering and Computer Science
University of Central Florida, FL
sarath.mannam13@knights.ucf.edu

Abstract

In this work, we will be using Detectron2, a codebase containing the implementation of state-of-the-art models in computer vision. We will train a custom object detection model using Faster R-CNN to carry out the task. A chess pieces dataset is used to classify the pieces in an image into respective classes. Our model aims to classify the chess pieces into respective classes and also drawing the bounding boxes around the same. Preliminary results, while not ideal, validate the concept and show that the model was capable of transforming the chess images into its respective types.

Introduction

Object detection has been an active research problem since the last decade. After the AlexNet Model that came up in 2012, by solving the ImageNet challenge using Neural Networks the desire to solve the Object Detection problem has accelerated. Region Based Network, R-CNN,[1] was presented at CVPR' 14, for the first time, where region proposals are used to detect Objects. Down the lane, improvements of R-CNN, namely, Fast R-CNN [2] at ICCV' 15 and Faster R-CNN [3] at NIPS' 15 were proposed. Faster R-CNN can be used towards real time object detection.

Detectron2 [4] is Facebook AI Research's library that provides state-of-the-art detection and segmentation algorithms. We are using Detectron2's Faster R-CNN X101-FPN model with Pretrained weights on Image Net and using Transfer Learning to train and build a model on our Chess Dataset. Using a ResNet+FPN backbone with standard conv and FC heads for mask and box prediction, respectively. It obtains the best speed/accuracy tradeoff.

Our chess dataset contains 693 bounding box annotated images. Each of the bounding box is named after a class that the box belongs to.

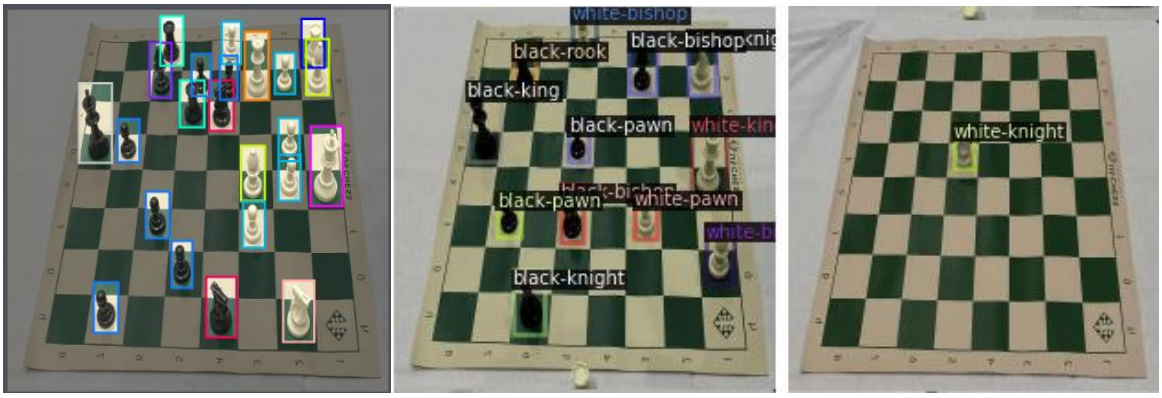

Figure 1: Images in the Dataset

A piece in the board can be named into black-bishop, black-king, black-night, black-pawn, black-rook, black queen, bishop, white-bishop, white-king, white-knight, white-pawn, white-queen and white-rook.

For the images contained in the dataset are augment by horizontal flip, hue between -20 to +20 and brightness between -15% to +15% adding a blur to the image. The images were 416 x 416. As a result, we had 693 images in our dataset. Data was split into 606 train (87%), 58 Valid (8%) and 29 Test (4%) Images.

Method

Faster R-CNN uses a Region Proposal network to generate the proposals where there is a possible object in the image. This RPN is an improvement over Fast R-CNN and R-CNN where both of these a region generator algorithm that was not learnt. R-CNN and Fast R-CNN generates regions using the Selective Search Algorithm. Here we use a Convnet to propose regions.

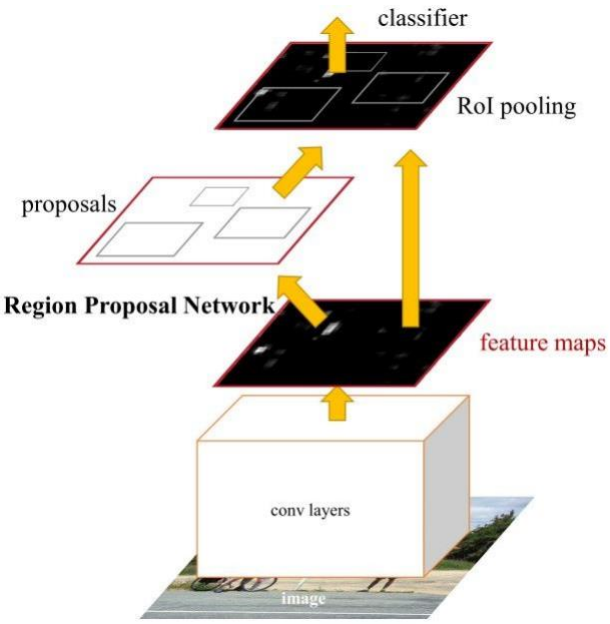

Figure 2 Overview of Faster R-CNN Algorithm

We are using Transfer learning to train our model. Our model has 13 classes and we change the last layer from the model we are using from Detectron2 Faster R-CNN to our needs. Our training parameters for this model are, config file is COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml with 4 workers and check point URL the same. We have a learning rate of 0.001 with max iterations 1500. Our batch size is 64 and the training happens on a Google Colab GPU Notebook. After initializing the configuration file of the model, we start training the model.

Results

We have achieved very good results on this dataset. Below is the results of this model using MS COCO Evaluation metrics.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.754
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.982
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.922
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.762
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.752
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.627
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.800
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.800
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.788
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.798
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = -1.000
```

Evaluation results of the bounding box are as follows:

AP	AP50	AP75	APs	APm	APl
75.357	98.249	92.225	76.189	75.175	nan

Per-Category Bounding Box AP as follows:

category	AP	category	AP	category	AP
pieces	nan	bishop	nan	black-bishop	59.318
black-king	84.705	black-knight	79.431	black-pawn	73.660
black-queen	82.931	black-rook	73.821	white-bishop	65.797
white-king	80.991	white-knight	78.319	white-pawn	77.081
white-queen	74.754	white-rook	73.483		

It is clearly evident that this model and faster-R-CNN Performed significantly well on the dataset and gave good results. Moreover, I intend to show the results using Tensorboard module and can be seen below.

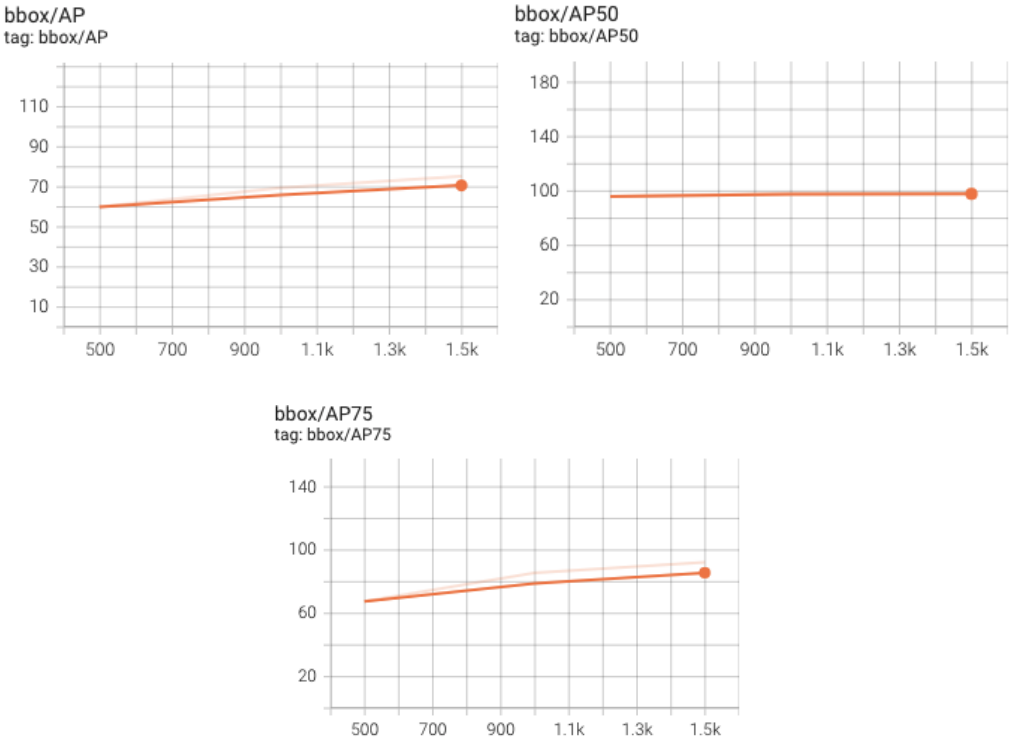

Figure 3 Results of AP for various threshold Values.

Total Loss and Bounding Box Regression Loss are also plotted and can be observed as

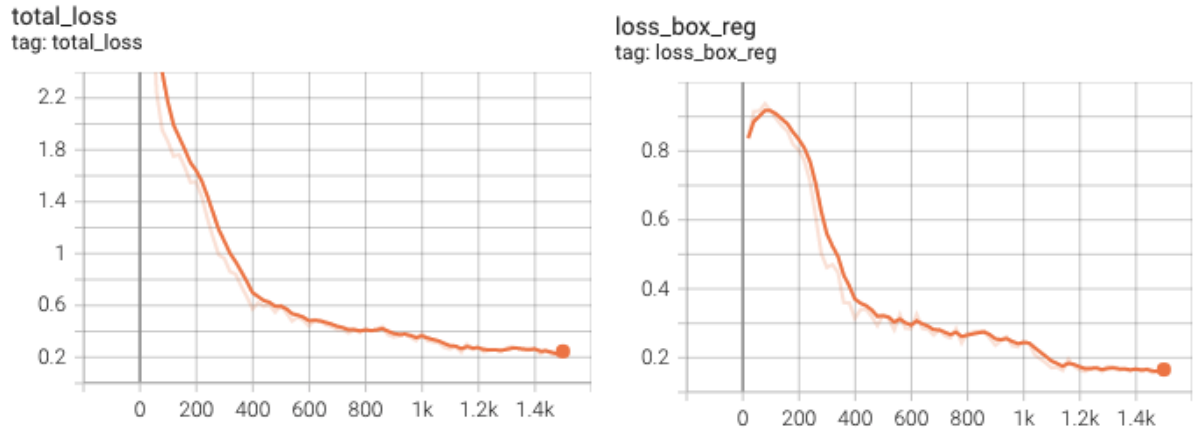

Figure 4: Total Loss and Loss Box Regression

We can further train the model for few more epoch for a better accuracy. Moving onto some of the results from the model after passing the test dataset.

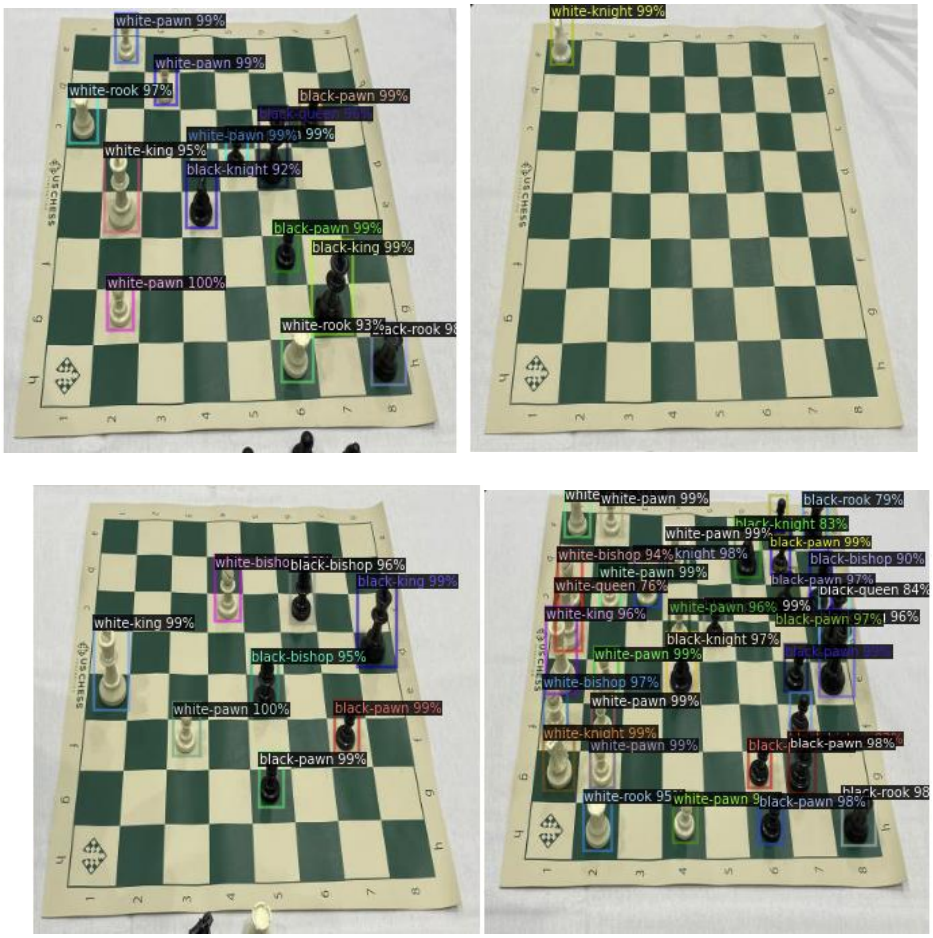

Figure 5: Results on the test set (a-e)

Discussion and analysis

We can clearly observe the efficiency of the model in classification by the bounding box it predicts and also the confidence with which it predicts. Starting from Fig 5(a) through Fig 5(e) the model did a good job of even detecting the farthest pieces from the board correctly and even the absence of chess pieces in Fig 5(e) were detected properly.

The loss for the model is getting small by each iteration and we can train the model for few more iterations if we want to decrease the loss further. AP for each class type was very satisfying and most of the classes have over 75 AP.

We can extend this model by using video clips of a match and track the movements visually. Also, we can increase the size of the dataset by choosing various colors of chess board and various size of chess pieces.

Conclusion

In this paper, we have seen how we can use a pretrained model of object detection and can use transfer learning to detect object of interest in our dataset. We have use pretrained Faster R-CNN and Detectron 2 to achieve the results we wanted.

References

[1] Rich feature hierarchies for accurate object detection and semantic segmentation | [CVPR' 14]

[2] Fast R-CNN | [ICCV' 15] |[pdf]

[3] [Faster R-CNN, RPN] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks | [NIPS' 15] |[pdf]

[4] Detectron2: Github https://github.com/facebookresearch/detectron2