YOLO for Chess Detection
An Implementation of the YOLO Algorithm

Miguel Conner, Ramón Talvi

March 2023

Abstract

In this project, we train a You Only Look Once (YOLO) model on a dataset of 292 images of chess peices with 2894 labels to make a real time object detector and classifier. We can predict on pictures, videos, and create a real time webcam output on our own chess pieces. We discuss the theory and limitations of these models, and some of the comparisons of the recent improvements on the original YOLO algorithm.

1 Introduction

Documenting every move in a tournament chess match is a tedious but necessary part of competition. However, using an real-time image detector and classifier would be a big step in potentially automating such a task. For this classifier to be suitable, it would have to make very few mistakes, be flexible and robust, and, above all, fast. We think the YOLO algorithm could be a good candidate for the first step of object detection in the documenting game moves.

The original YOLO algorithm was a landmark moment in the world of computer vision when it came out in 2015; it was a novel way of doing real-time object detection and classification that absolutely smashed the competition (Redmon, Divvala, et al. 2016). At the time, the best performing networks were Deformable Part Models (DPM) and Region Based Convolutional Neural Networks (R-CNN). DPMs do object detection using a sliding window approach, running a classifier on many different parts of an image. R-CNNs extract Regions of Interest (rectangles that encapsulate objects) from an image and run those through a Support Vector Machine to predict the class of an object. However, DPMs classified very poorly and R-CNNs took over 40 seconds process a single image, making both techniques less-than-ideal candidates for real-time object detection. R-CNNs were streamlined to shave a few seconds off the run-time at the expense of accuracy, but it took the completely new architecture of the YOLO algorithm to make real-time object detection and classification a reality.

YOLOv1 process images at 45 frames per second with an accuracy comparable to that of R-CNNs. It has since been improved in successive versions that improve both run-time and accuracy. A few notable improvements are YOLOv2 and YOLOv7. YOLOv2 adds features such as anchor boxes, batch normalization, among others; and combines that with a hierarchical classifier (WordTree) that allows for training and detection of a much larger class size (Redmon and Farhadi 2016). YOLOv7 uses model reparamaterization, model scaling, and an improved architecture to simplify the backpropagation of gradients, leading to a much faster and more accurate model(Wang, Bochkovskiy, and Liao 2022).

2 Data

The data set was downloaded from Roboflow, and is composed of 292 images of size $416x \times 416$, with a total of 2894 labels (Roboflow 2021). The labels spanned a total of 12 different classes: white-king, white-queen, white-bishop, white-knight, white-rook, white-pawn, black-king, black-queen, black-bishop, black-knight, black-rook, and black-pawn. Finally, the some images were duplicated and rotated by 90 or 180 degrees to create a more robust detector.

3 Methods

The YOLO algorithm uses regression to detect and classify objects using a novel detection technique. It is a supervised learning algorithm; it takes images with labeled with boxes around each object as well as information about the class of the object, to train the algorithm. Then, given an image without any labels, it will attempt to detect the objects in the image and classify them. It is based on a system of grid cells and a CNN architecture that are discussed in more detail below. Improvements on the original paper algorithm are also discussed.

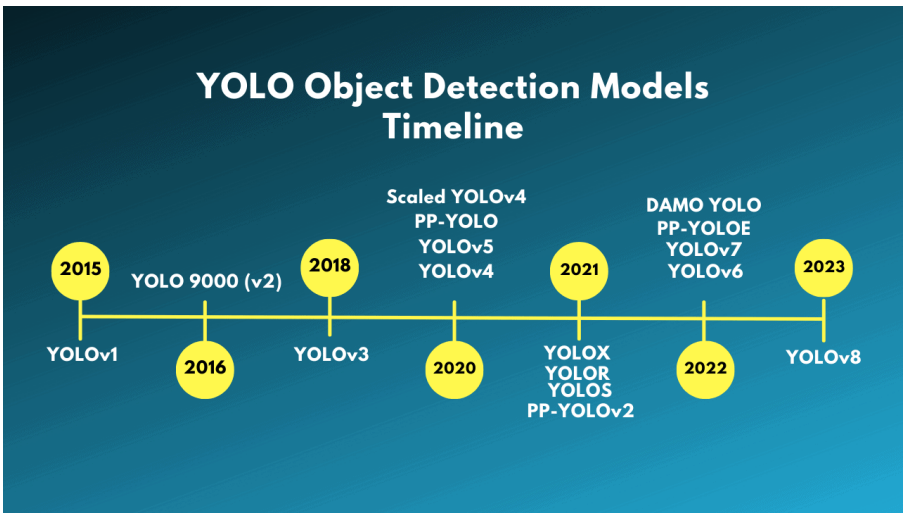
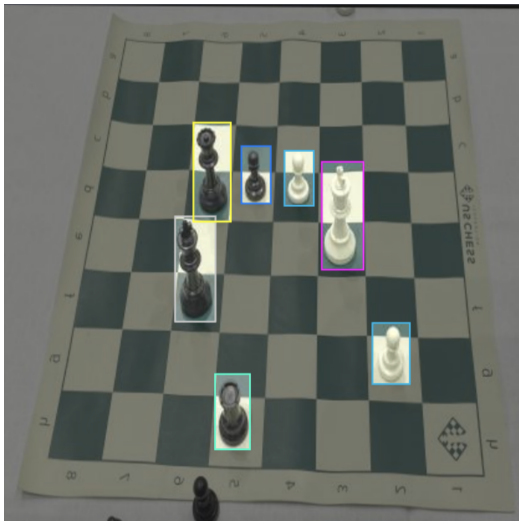


Figure 1: Timeline showing the evolution of the YOLO algorithms, from (Rath 2023).

(a) An example of an image used for training that does not have labels.



(b) An example of a training image with the labels visualized with colored boxes.

Figure 2: Two images from the dataset, one with labels and one without. The labels are included in a seperate .txt file, and are shown here as colored boxes only for the purpose of visualization.

3.1 YOLO

3.1.1 Grid Cells and Bounding Boxes

The grid cell/bounding box structure of the YOLO algorithm allows for efficient detection of objects all over the image. The image is broken up into a grid of S by S cells. Each cell will predict B bounding boxes–boxes that trace an outline around each object we would like to classify. If the center of the bounding box falls into a particular cell, then it is the responsibility of that cell to classify that object.

The objective is to output only the bounding boxes that have a high probability of being correct, so we define the concept of the confidence score for each bounding box. This score is the product of the conditional class probability, the probability of the object, and the IOU.

$$\Pr(Class_i|Object) * \Pr(Object) * \text{IOU} \tag{1}$$

We train our model on C classes. The Intersection Over Union (IOU) measures how much overlap there is between the predicted box and the ground truth. In the end we want to output vectors with coordinates and dimensions of bounding boxes, as well as the classification. This produces a $S \times S \times (B * 5 + C)$ tensor as the output. The grid cells and bounding boxes are visualized in Fig. (3).

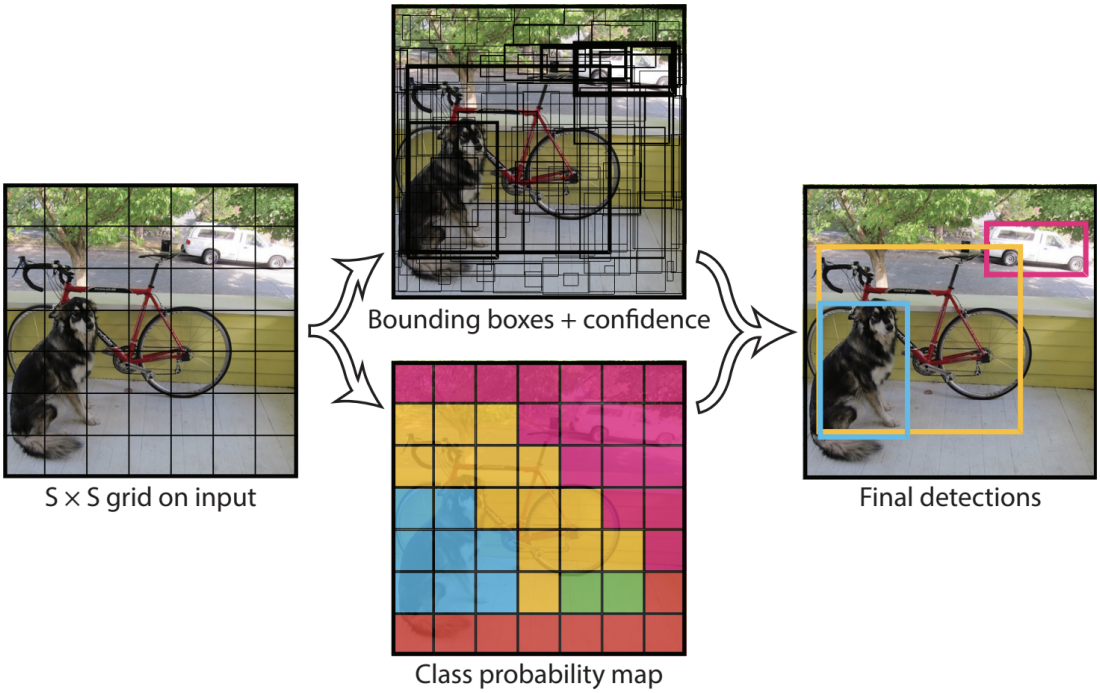


Figure 3: This figures describes how the grid cells (defined by an $S \times S$ grid) and the bounding boxes (left) interact to output classified bounding boxes (right). The middle two steps describe the calculation of probabilities for each grid cell and the assignment of confidence for each bounding box. Only bounding boxes above a certain threshold (set by the user) are outputted. (Redmon, Divvala, et al. 2016)

3.1.2 Architecture

The architecture of the YOLO algorithm (shown in Fig. (4)) is a neural network architecture composed of 24 convolutional layers and 2 fully connected layers. Between each layer there is a leaky rectified linear unit activation function.

The network is then optimized using a sum of squares loss function, that is chosen because it is relatively easy to optimize. The terms of the loss function are balanced with parameters so as to ensure that bounding boxes of different sizes are not penalized differently.

Other notable components of the architecture are dropout (dropout rate of 0.5), random image dimension adjustment by up to 20%, and random adjustment of the color space. These changes make the algorithm more robust and less likely to overfit.

3.2 YOLOv2 and YOLO9000

The second version of the YOLO algorithm improves the speed and accuracy of the algorithm by making a few changes: adding batch normalization, anchor boxes, increasing the resolution of the images in training, making a few tweaks to the architecture, and others Redmon and Farhadi 2016.

Batch normalization is a technique used to standardize data that involves normalizing batches of data within the network. Anchor boxes are bounding boxes with fixed ratios that we use as templates for the predicted bounding boxes to output. Using these we can remove one of the fully connected layers from the network.

In the same paper, the authors introduced YOLO9000, a modified YOLOv2 algorithm has been trained on two different datasets (COCO and original ImageNet) in a way that it is able to make detections of objects even for which it does not have any data. This is done through the implementation of a hierarchical word classifier called WordNet, which relates words like “dog,” “canine,” and “Yorkshire Terrier.” This allows for a much more expressive object classifier.

3.3 YOLOv7 and YOLOv8

Given we implemented bith YOLOv7 and YOLOv8 to asses how the state-of-the-art models performed, we will make some brief review over the mayor innovations before diving into the results of the trained algorithms.

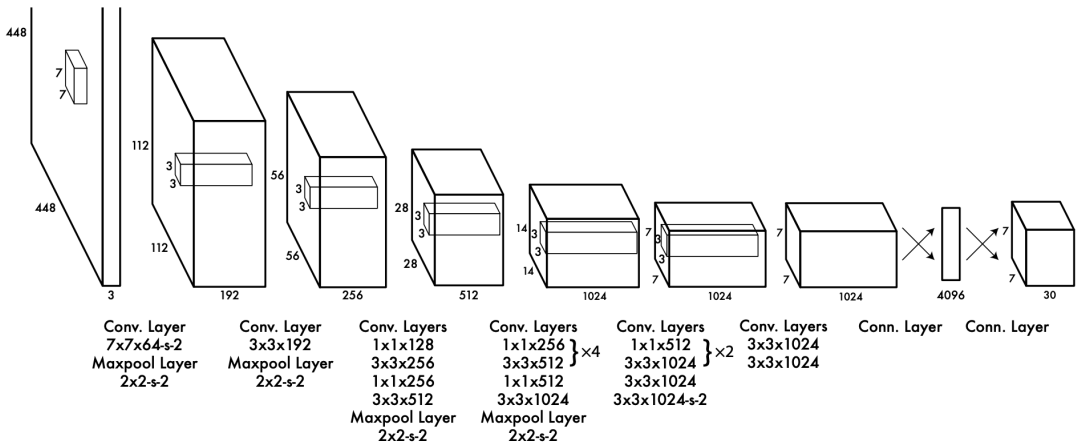


Figure 4: The architecture of the original implementation of the YOLO algorithm. It combines convolutional layers with connecting layers to produce a tensor of predictions four bounding boxes. Here the parameters are set to $S = 7$ (7×7 grid cells), $B = 2$, and $C = 20$. (Redmon, Divvala, et al. 2016)

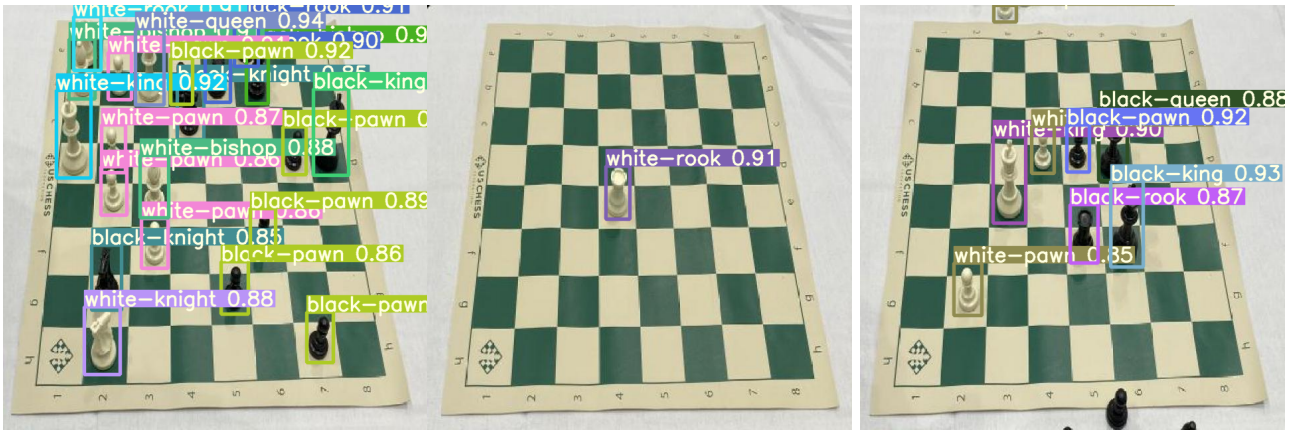


Figure 5: Three images from our test data set run through our YOLOv7 model.

YOLOv7 improved on previous versions by allowing the model to handle images of different sizes and aspect ratios, which makes object detection more accurate and reliable. It also used a new technique that combines different approaches to object detection, leading to more precise results (Wang, Bochkovskiy, and Liao 2022).

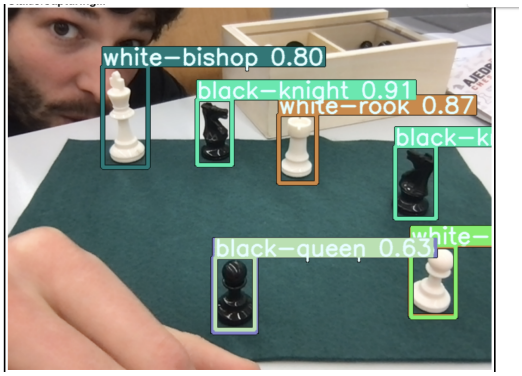
In YOLOv8, they added self-attention modules that help the model focus on important parts of the image and ignore the rest, which makes it faster and more accurate. They also introduced a new technique that makes it better at detecting small objects and reducing false negatives. Finally, they added a feature that helps the model capture more information about objects of different sizes and shapes (Rath 2023.

4 Results and Discussion

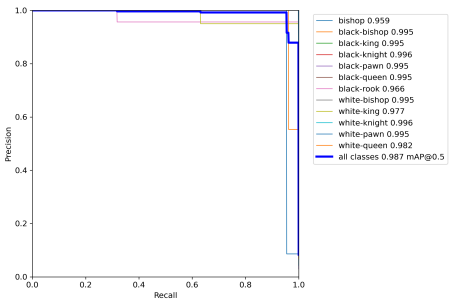
4.1 YOLOv7

We successfully trained network to detect and classify the target data. Our algorithm was cloned from the official YOLOv7 implementation from Github (WongKinYiu n.d.). Our network was trained with a batch size of 16 and showed great results in 50 epochs. We trained our model in Google Colab, and it took about and hour to run. Three examples of outputs of our network on images from test set are shown in Fig. (5), and a screenshot of the real-time webcam object detector was is shown in Fig. (6a). The PR curve on a test set is shown in Fig. (8). Sample pictures and videos, including a screen recording of the real-time detector, will be submitted with this report.

Our model showed accurate classification of the objects in the test sets, as can be seen by the impressive PR curve and a visual inspection of the example images in Fig. (5). We note that labels are both accurate and given with high degrees of confidence. When running our network on our own chess pieces that look quite similar, we note that our results are not as impressive. This will be further discussed in the section labeled Limitations of Our Model below.



(a) A screenshot from the output of our real-time webcam detector. All but one piece are classified correctly.



(b) The PR curve for our model after 49 epochs.

Figure 6: More results from the YOLOv7 implementation.

Figure 7: Two images from our test data set run through our YOLOv8 model.

4.2 YOLOv8

After training the YOLOv7 model on our chess data we implemented the YOLOv8 algorithm aslo trained on our chest dataset. The intention was to attempt to train the state-of-the-art YOLO model.

The algorithm was also cloned from the official YOLOv8 implementation on Github, and the training was conducted with a batch size of 20 over 100 epochs using Google Colab. The efficacy of our model was evaluated on a test set and the detection outputs for two sample images on our test set as illustrated in Fig.(7). Moreover, we also show the precision-recall (PR) curve in Fig. (8).

Overall, our model achieved high accuracy in classifying and identifying the different chess figures. This is suggested by the in-sample metric provided by the precision-recall curve and confirmed by the labelling the model provides when faced when unseen test data. It is worth noting that the labels were assigned with high confidence levels. We do not detect substantial divergence in the results attained by YOLOv8 with respect to YOLOv7. Despite our model seems to generalize relatively well to unseen data, it has some limitations that are shared with the YOLOv7 and will pointed out next.

4.3 Limitations of Our Model

We hoped our data set would be enough for our model to generalize well to our own chess pieces, and though we found that our model was able to correctly classify chess pieces in many situations, it also failed to perform as well as in our test environment. This was very apparent in the real-time detector we built, were we saw many type 1 and type 2 classification errors (as seen in Fig. (6a); the black pawn is classified as a black queen, for example). We suspect this is due to the lack of diversity in our test data. After all, the images in our training set were shot with the same chess pieces on the same chess board with the same lighting from a similar angle. We suspect that by adding more pictures in different environments (that we would have to manually label), we would have made our predictions more robust.

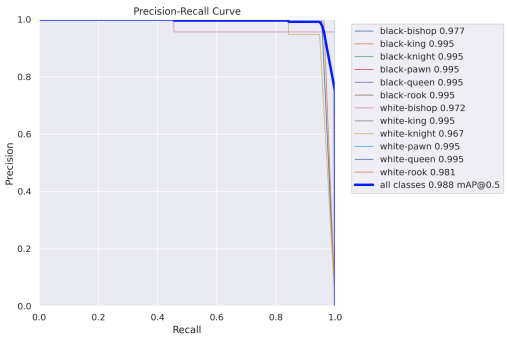


Figure 8: Our Precision-Recall Curve of trained YOLOv8 model

5 Conclusion

In this project, we implement two YOLO algorithms on a dataset of chess pieces. We train the models using 50 and 100 epochs respectively to achieve accurate results on our validation sets. Both modles worked quite well, and classified the chess pieces with surprising accuracy. Our models did not generalize as well to custom situations as they did when applying the model to the test data, but we believe this is due to the limited diversity in the training data.

Further work could improve upon the accuracy of the classifier by including more varied images in the training and test set. In addition, this project could be combined with some software that tracks the position of the pieces as well, and could be used to record moves in tournament play.

References

Redmon, Joseph, Santosh Divvala, et al. (June 2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Redmon, Joseph and Ali Farhadi (2016). *YOLO9000: Better, Faster, Stronger*. DOI: 10.48550/ARXIV.1612.08242. URL: https://arxiv.org/abs/1612.08242.

Roboflow (Apr. 2021). *Chess Pieces Object Detection Dataset*. URL: https://public.roboflow.com/object-detection/chess-full.

Wang, Chien-Yao, Alexey Bochkovskiy, and Hong-Yuan Mark Liao (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. DOI: 10.48550/ARXIV.2207.02696. URL: https://arxiv.org/abs/2207.02696.

Rath, Sovit (Mar. 2023). *Yolov8 ultralytics: State-of-the-art yolo models*. URL: https://learnopencv.com/ultralytics-yolov8/.

WongKinYiu (n.d.). *Wongkinyiu/Yolov7: Implementation of paper - yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. URL: https://github.com/WongKinYiu/yolov7.