Machine Learning and Object Detection in Soccer

Maggie Du

Abstract

In recent years, the use of artificial intelligence has been growing in a variety of fields, helping humans perform tasks with greater precision. In soccer, artificial intelligence has been gradually incorporated through systems such as goal line technology and Video Assistant Refereeing. Such technologies reduce the risk of human errors that may prove to be unfair and/or inefficient due to refereeing tasks causing too much delay. This research explores a possible machine learning model, YOLO, that can process datasets of videos and images from soccer matches more quickly and accurately. Existing image datasets from previous soccer matches were analyzed using a neural network to process and detect where such important objects as the ball or players are. Using the YOLO object recognition algorithm, I coded a program that outputs the precision and recall of the YOLO object detection algorithm using the true positive, true negative, false positive, and false negative counts. Using these values I generated a precision versus recall curve by sweeping through different object detection confidence values. As expected, precision decreased as recall increased, but the AUC is fairly high (around 0.812), comparable in AUC to prior work using VGG16, but with much quicker completion time. Although the default training of YOLO is a good starting point, by diving deeper into training YOLO specifically for soccer images, we could compare and contrast all the possibilities and perhaps find one that proves to be the most efficient and accurate.

I. Introduction

The game of soccer (football) is undoubtedly one of the most popular sports around the world. The annual UEFA Champions League tournament attracts hundreds of millions of viewers across its season, while the quadrennial World Cup now attracts billions of viewers from all across the world. However, the game often comes to the finest of margins: whether or not a ball has fully gone over the goal line or an attacking player has just barely crossed into an offside position determines if a potentially game changing goal can stand, especially because of how low scoring most games tend to be. Thus, the task of object detection comes into play significantly, as we must be able to accurately detect the location of a player or ball relative to other indicators during a match. Using artificial intelligence has proven to be very useful in the past for such uses, especially for VAR technology (video assistant refereeing). However, with the need for a whole team of assistant referees still prominent in most games, it is clear that there is room for improvement and the possibility of more resource and cost efficient ways to reach the same goal.

One common and increasingly powerful tool for quick object detection is YOLO, or You Only Look Once. YOLO uses a Convolutional Neural Network (CNN) to learn different features/ classifications through a training process, allowing it to be able to differentiate between a huge variety of objects ranging from vehicles, clothing items, people, etc. YOLO is special from other detection algorithms in that it is able to view the entire image at once, meaning it makes its classifications quickly but still relatively accurately. The algorithm recognizes 80 different object classes and can place bounding boxes around each detected object, along with a confidence

value in the form of a percentage revealing how confident the model is in its classification.This project uses YOLO v3, released in 2018, which has improvements in speed, precision for objects of all size ranges, and class specification over the original YOLO algorithm.

The overarching question being answered is: can YOLO reliably and accurately identify where or if there is a soccer ball in an image? Thus, the goal of the following procedures is to test the overall accuracy of this model in detecting the ball in snapshots of soccer matches. Precision and recall will be the performance measures used to evaluate the model: precision being how much of the classifications made are correct, and recall being how many of the total objects in question are correctly detected.

II. Methods

Dataset

The dataset I used is from the Islamic Azad University Football Dataset (IAUFD), made specifically for soccer image analysis. It contains 100k images in total of different snapshots captured in drone/ bird's eye view from previous soccer games. This dataset specifically was chosen because of how many images it contained, and due to the fact that its data was labeled by class, providing a huge amount of data to be used and analyzed in testing the YOLO algorithm. Its images were also quite complex and from real games, giving a realistic assessment of the model's abilities in identifying a soccer ball in any position on the field.

Figure 1: Example image from the IAUFD 100k image data set depicting a moment in a Wales vs Northern Ireland match, including one soccer ball and several players on the pitch. This image would be labeled with a 1 in the ball class and a 1 in the referee class.

Obtaining classification results

I began my research by using the existing dataset of soccer images from previous matches,

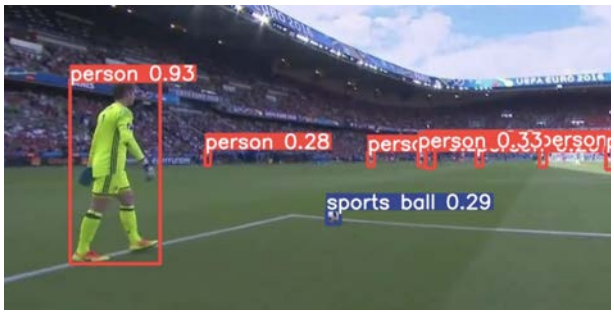
then uploading these to a YOLO v3 colab. Running through every image in the repository produced a list of identified objects and the frequency for every object type.

```
10.jpg: 736x1280 54 persons, 1 handbag, 5 sports balls, 2 baseball gloves, 24 tennis rackets, Done. (0.077s)
100.jpg: 736x1280 3 persons, 1 frisbee, 1 baseball bat, Done. (0.075s)
101.jpg: 736x1280 10 persons, Done. (0.078s)
102.jpg: 736x1280 3 persons, 1 bench, 2 cell phones, Done. (0.077s)
103.jpg: 736x1280 5 persons, Done. (0.077s)
```

Figure 2: YOLO v3 classification results, for 5 different images, depicting the image size in pixels, types of objects detected, frequency of each class, and run time in seconds. In our case, the only class of interest is the sports ball class, detected in image 10.jpg.

Calculation of TP, FP, TN, FN values

Initially, I manually calculated the true positive, false positive, true negative, and false negative values for every image result by comparing what I saw in the inputted image against the results of detected sports balls by the YOLO v3 algorithm. A true positive occurs when the algorithm detects and classifies an object correctly, a false positive occurs when the algorithm detects an object but classifies it incorrectly, a true negative occurs when the algorithm does not detect any object in an image that does not contain any of that object type, and a false negative occurs when the algorithm does not detect any object even when the image does contain one.



True Positive: the ball was detected correctly

False Negative: the ball was not picked up by YOLO

False Positive: YOLO falsely detected a

True Negative: YOLO correctly detected

ball at the bottom of the image	*that there was no ball in the image*

Figure 3: Table depicting TP, FN, FP, and TN results by YOLO v3 algorithm.

While this produced accurate answers, the process was highly inefficient and needed to be automated in some way. Thus, I coded a program in JAVA to automatically calculate the TP, FP, TN, and FN values by reading the YOLO output text file and comparing it to an "answer key" of the actual contents in each image, derived from the previously labeled classes in the image dataset.

```
if(ball_image[i] == 0) //if there is no ball in the image
{
    if(ball_yolo[i] == 0) TN ++; //if YOLO didn't detect anything add to true negative
    else FP ++; //if YOLO detected a ball add to false positive
}
else //if there is a ball in the image
{
    if(ball_yolo[i] == 0)  FN ++; //if YOLO didn't detect anything add to false negative
    else TP ++; //if YOLO detected a ball add to true positive
}
```

Figure 4: Logic for determining true positives, true negatives, etc: if there is no sports ball in the image, the classification would count as a FP if a sports ball was detected by YOLO, or a TN if a sports ball was not detected by YOLO. If there is a sports ball in the image, the classification would count as a TP if a sports ball was detected by YOLO, or a FN if a sports ball was not detected by YOLO.

Finding Average Precision and Recall

Precision is a measure of how many of the classified objects detected by YOLO are actually a soccer ball, in other words, how much we can trust the algorithm's choices. So, dividing TP by TP + FP gives precision in decimal form (1), as TP + FP is the total number of objects the algorithm claims to be a soccer ball, and TP is just the ones that have been labeled correctly.

Recall is a measure of how many of the total soccer balls have been detected by YOLO. So, dividing TP over TP + FN gives recall in decimal form (2). TP + FN is the total number of soccer balls actually in the dataset of images, as TP are the ones correctly labeled and FN are the ones that have been missed by the algorithm.

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

Because of how the two values are defined, in theory, precision and recall should have an inverse relationship with each other: as precision of an object detection model increases, its recall generally decreases, and as recall increases, precision generally decreases. This should be quite intuitive, since a model that has high precision must be very confident in each of its

decisions before classifying an object into a certain group. This, consequently, means that many of the other objects in the image will be missed if the model is not confident enough in it. On the other hand, a model that has high recall must be able to capture almost all of the objects in the image, meaning that some of its detections are bound to not be relevant/ what we are looking for.

Sweeping Through Varying Confidence Levels

Both precision and recall values depend on confidence levels, which dictate how likely the boundary box in question contains an object of interest. For example, if the box is labeled "sports ball 0.78", then that means the model is 78% confident that there is a sports ball within that box. The confidence parameter set in the run line determines how confident YOLO must be in order to visually mark the box as a detected object. Thus, it can be inferred that the higher the required confidence level is, the greater the precision will be, because the algorithm is close to certain that each detected object is correct. Consequently, the lower the required confidence level is, the greater the recall will be, because it is likely that all possible objects will be picked up, even if the confidence level is low.

To sweep through all confidence levels, I started from a confidence level of 0.00 and worked up to a confidence level of 0.80 by intervals of 0.01, 0.05, and 0.10, stopping at 0.8 because YOLO fails to detect any soccer balls after confidence levels above that point. I then calculated the precision and recall values for each level, eventually producing a precision vs recall curve.

Confidence	Recall	Precision
0.00	1.000	0.472
0.01	0.987	0.576
0.02	0.935	0.632
0.03	0.896	0.645
0.04	0.818	0.656
0.05	0.727	0.675

Figure 5: Condensed table of confidence levels ranging from 0.00 to 0.05, along with their subsequent recall and precision values as calculated by previous steps.

III. Results and Conclusion

The final precision vs recall curve demonstrated the predicted inverse relationship between the two values: as recall increased in value, precision decreased in value. The AUC (area under the curve) was around 0.812, relatively high considering that the max possible value is 1.00, which denotes a perfect model. The larger the AUC, the better the model is, because it means that as recall increases and gets closer and closer to 1, the precision still remains at a relatively constant value and does not drop dramatically. Thus, the YOLO v3 model can be considered very effective for object detection in soccer matches.

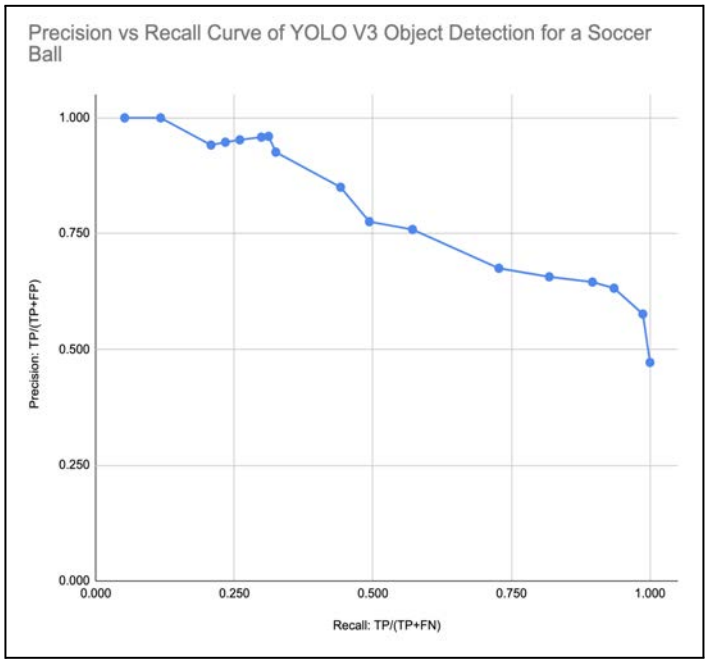


Figure 6: Precision vs recall curve with a max precision value of 1.00 when recall is ~0.05, dropping to 0.47 when recall is 1.00. Precision decreases at a relatively steady rate as recall is increasing, but faces a sharp drop once recall values near 0.94. The estimated AUC is 0.812.

IV. Discussion

While this research proves that YOLO has the potential to accurately detect and locate a soccer ball in the middle of a match, there are still many questions to be asked about how much of a benefit it can truly be to the game. Compared to a human referee, it still makes mistakes by either missing the object in question or detecting a ball where there isn't one. Exactly how precise the algorithm needs to be for use in professional games is debatable, but if used in conjunction with human referees as a double check, it may bring significant improvements in efficiency.

Another important consideration is which takes precedence over the other: precision or recall? In the context of sports and soccer, recall is slightly more important both to the professional leagues and to the fans. If there is a ball or player gone unnoticed, there could be huge repercussions for both teams, as a goal or an offside call could be disregarded. Thus, having higher recall, even if that means falsely detecting certain objects across the field, may be acceptable until better systems are developed. If the algorithm were to incorrectly locate a ball where there is none, a human co-referee could easily notice such a mistake and make the proper call necessary, while still taking advantage of a machine learning model that can more quickly find the exact location of the actual ball.

Future work can be conducted on training YOLO specifically for the task of analyzing soccer matches, and seeing how that can improve its precision and recall.

References

1. Cioppa, Anthony, et al. “Soccernet-Tracking: Multiple Object Tracking Dataset and Benchmark in Soccer Videos.” *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2022, https://doi.org/10.1109/cvprw56347.2022.00393.

2. “Image/Video Analysis - IAUFD.” *Google Sites*, 2021, https://sites.google.com/view/image-and-video-analysis/iaufd?authuser=0.

3. Meel, Vidushi. “Yolov3: Real-Time Object Detection Algorithm (Guide).” *Viso.ai*, 2 Jan. 2023, https://viso.ai/deep-learning/yolov3-overview/#:~:text=YOLOv3%20AI%20models-,Wa%20is%20YOLOv3%3F,network%20to%20detect%20an%20object.

4. Saha, Sumit. “A Comprehensive Guide to Convolutional Neural Networks-the eli5 Way.” Medium, Towards Data Science, 16 Nov. 2022, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-network-the-eli5-way-3bd2b1164a53.

5. Ultralytics. “Ultralytics/yolov3: Yolov3 in PyTorch.” GitHub, 2018, https://github.com/ultralytics/yolov3.

6. Zanganeh, Amirhosein, et al. “Institution of Engineering and Technology - Wiley Online Library.” *Islamic Azad University Football Dataset*, https://ietresearch.onlinelibrary.wiley.com/.