Comparing Algorithms for Detecting People

Daniel Lindholm, Jacob Borg,
Nikolaj Thorsen Nielsen

December 15, 2020

Abstract

Detecting people in a video in real time requires you to run an object detection algorithm on different images multiple times a second. This article provides a comparison of three different object detection algorithms, comparing speed and accuracy. Our testing shows that YOLO is the best of the chosen algorithms, when looking at accuracy and the tests show that it is on average 6 times faster. Using YOLO makes real time object detection in a video possible without losing frames per second.

1 The Problem

In order to be able to do object tracking in real time on video, a fast algorithm is needed to process multiple images per second, and an algorithm that is good enough at detecting the needed object, in our case, a person. These two characteristics are important since they are needed in order to reliably track a moving object.

2 Benchmark

All the benchmarks are made on the same computer with an intel i5-4460 at 3.20 GHz and 8 GB ram, running the different algorithms on the same images and are all written in Python. The times have been calculated using pythons time.time()

3 Alternatives

3.1 Faster R-CNN (Region-Based Convolutional Neural Network)

Faster R-CNN works by using selective search to extract regions of interest in a picture, and then resizes these regions to a specific size, and runs these sub-images through a trained neural network to see if a known object can be found on any of them [4, sec. 2].

```
def get_prediction(self, img, transform):
        img = transform(img)
        pred = self.model([img])
        return pred[0]
```

3.2 YOLO (You only look once)

What makes YOLO different from other algorithms in that it is only passed through a neural network once, and then the result is processed by a non-max suppression algorithm that makes sure the same object hasn't been found multiple times [1].

```
def get_prediction(self, img):
        bboxs, labels, confs = cv.detect_common_objects(img)
    return bboxs, labels, confs
```

3.3 HOG (Histogram of oriented gradients)

HOG is used in computer vision, as a feature descriptor for object detection. First a gradient orientation is calculated for small areas of the image, then HOG is able to extract features from the gradient orientation data, this is done by HOG being able to detect edges and the direction of the edges [2].

```
def get_prediction(self, image):
        if image.shape[1] < 400:
                (height, width) = image.shape[:2]
                ratio = width / float(width)
                image = cv2.resize(image, (400, width * ratio))
        img_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        rects, weights = self.hog.detectMultiScale(img_gray,
                                        winStride=(2, 2),
                                        padding=(10, 10),
                                        scale=1.02)
        return rects, weights
```

4 Findings

Figure 1 shows that YOLO consistently is the fastest of the three algorithms across all the sample images. Figure 2 shows that YOLO is consistent in its runtime, only one data point is a significant outlier. The outlier might be due to loading the model on the first run. This is also seen on table 1 that even though YOLO has a max of 4.441 seconds the interquartile range is just 0.132 seconds and the standard deviation is 0.17 seconds.
The data points for the HOG algorithm has the widest range, as seen in table 1 it has an interquartile range of 7.823 seconds and a standard deviation of 3.866 seconds. We can see on figure 3 that there is a gap in the middle with no data points. The gap in the data is due to the difference in the sizes of the images, the smaller the image, the faster it can process the image.
Faster R-CNN is consistent in its runtime, as can be seen on figure 4, it has an

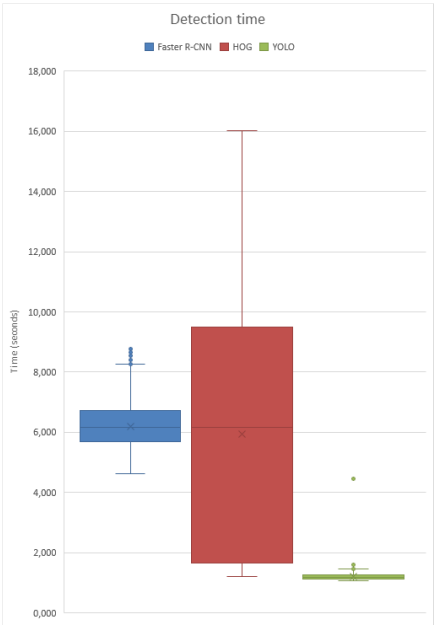

Figure 1: Boxplot over the time it took the three algorithms to finish person detecting. Each of the algorithms has a data size of 600 data points, which has been record from processing 6 different images of varying size and motive.

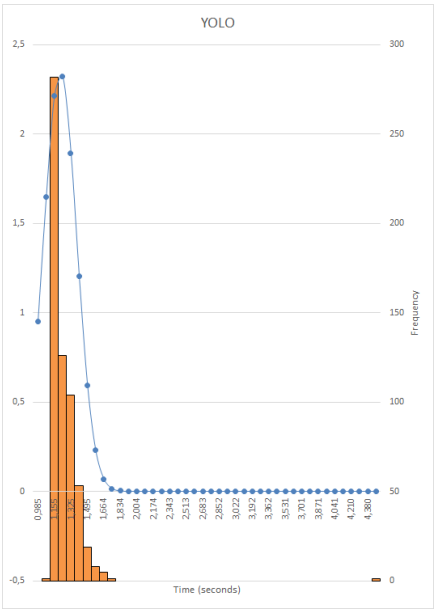

Figure 2: Histogram and Bell Curve for the data collected from running YOLO

	Faster R-CNN	HOG	YOLO
Min	4.610	1.196	1.070
25-percentile	5.682	1.658	1.130
Median	6.166	6.172	1.161
75-percentile	6.705	9.481	1.262
Max	8.772	16.026	4.441
Interquartile Range	1.023	7.823	0.132
Mean	6.178	5.929	1.213
Standard Deviation.	0.873	3.866	0.170

Table 1: Statistical variables calculated to make boxplot in figure 1. Time (seconds).

interquartile range of 1.023 seconds and a standard deviation of 0.873 seconds, but is still considerably slower than YOLO, almost 6 times slower.
The images used for testing contain 13 people. Table 2 shows that Faster R-CNN found all but also one false positive. YOLO missed one person but it did not get any false positives. HOG got 5 people and detected the legs of one person therefore the .5 point, it also got a false positive.

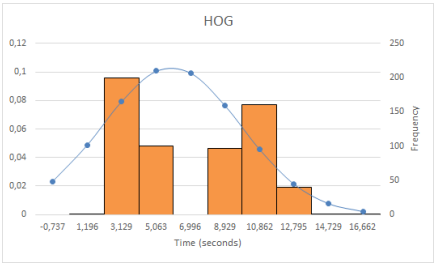

Figure 3: Histogram and Bell Curve for the data collected from running HOG

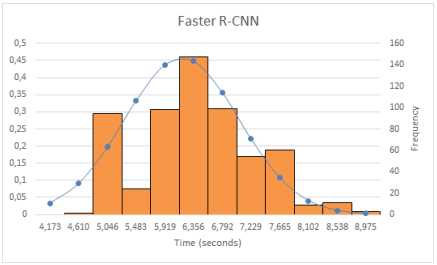

Figure 4: Histogram and Bell Curve for the data collected from running Faster R-CNN

	Faster R-CNN	HOG	YOLO
Correct Detections	13	5.5	12
False Positives	1	1	0

Table 2: On the 6 images we tested, there were a total of 13 people, this describes how many people the algorithm found out of those with over 90% confidence.

5 Conclusion

YOLO is by far the fastest of the different algorithms we tested. So if speed is the primary factor YOLO would be a great choice. HOG is both slower and less precise, it is better for finding shapes, but because people can have different stances and angles, it makes it very difficult for this approach. Faster R-CNN was the only algorithm that found all the people in the pictures, but it did however also have a false-positive. So if speed is not an issue and completeness is the most important thing, Faster R-CNN is a great choice.
Overall for tracking people in a stream of images, we found that YOLO's speed and accuracy was the best fit, as it didn't have any false positives, and also was on average 6 times faster than Faster R-CNN. This makes YOLO soundness, from our experiment.

6 Discussion and Future Work

The current code is run on the CPU, in the future it could be interesting to run some of these algorithms on a dedicated graphics card, for example you can use a newer nVidia card with integrated CUDA cores to speed up the processing time by as much as 380% faster[3].
Some of the different techniques these selected algorithms can be run using other weights that can make them faster, but less precise.

References

[1] Overview of the yolo object detection algorithm. `https://medium.com/@ODSC/`

overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0, 2018 (accessed December 10, 2020).

[2] Feature engineering for images: A valuable introduction to the hog feature descriptor. https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/, 2019 (accessed December 10, 2020).

[3] Opencv 'dnn' with nvidia gpus: 1549 https://www.pyimagesearch.com/2020/02/10/opencv-dnn-with-nvidia-gpus-1549-faster-yolo-ssd-and-mask-r-cnn/, 2020 (accessed December 11, 2020).

[4] Pulkit Sharma. A step-by-step introduction to the basic object detection algorithms (part 1). https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/, 2018 (accessed December 10, 2020).

List of Figures

List of Tables