Applying YOLOv5 to the Failure Dataset

Michael Lee
ml3406@rit.edu

Owen Shriver
ofs9424@rit.edu

Abstract

In this project, we applied YOLOv5 object detection algorithm to the Failure dataset. In our first experiment, we applied the algorithm to the failure frame of each video. We found a maximum object detection rate of 83%, a peak accuracy rate of 92%, and a peak combined correct classification rate of 66%. In the second experiment, we took 10 frames from every selected video and compared the results with those acquired using only the failure frame. We discovered an increase of 7% in the object detection rate and an increase of 2% in the accuracy.

1. Introduction

The Failure dataset is a dataset put together specifically for this class (CSCI 631: Foundations of Computer Vision) and annotated by its students over the course of the semester. It contains a set of videos, each containing some kind of failure, annotated with background information and object bounding boxes at the start, end, and failure frames. YOLOv5 is the latest version of the YOLO algorithm, standing for You Only Look Once, which detects and classifies objects in a single step [1]. In this project, we applied YOLO to the annotated failure frames and analyzed the results.

2. Related Work

A paper about the current version of YOLO (YOLOv5) has not yet been released, however previous versions have the same underlying structure. YOLO was designed to be a fast, lightweight object detection system [1]. It was trained on the COCO dataset, which contains objects of 90 different classes representing everyday objects, like people, animals, and vehicles. An iconic view is where the object is in profile, centered, and not obstructed by other objects in the scene; these conditions are ideal for object recognition, but they are not always realistic. The COCO dataset contains images in non-iconic views, and its goal is to facilitate recognition in situations that are not ideal, but rather more representative of everyday situations [2].

The Failure dataset videos were taken from a variety of sources, many of them from social media and shot on mobile phones. These videos contain non-ideal angles and lots of objects, so we hypothesized that an algorithm trained on the COCO dataset would be effective.

3. Method

We ran our experiments using the YOLOv5s (Small) model, which was designed to be the most lightweight form of YOLO with only 7.3 million parameters. We implemented two separate experiments using this model. In our first experiment, we simply used the failure frame from each video as the input to the network, and observed the resulting detections. We evaluated the performance of the model using several different metrics (see below). Since our dataset are frames from videos, we decided to add temporal information to the model in the second experiment. We first randomly selected 22 videos from the dataset and annotated extra 9 frames for every video (4 frames right before the failure frame and 5 frames right after it). After that, we implemented single-frame detection and multi-frame detection to compare their performances. In single-frame detection, the failure frame from every selected video was used; in multi-frame detection, the failure frame and the extra-annotated frames from every selected video were use. If an object was detected in several frames, we took the average as its result for evaluation.

3.1. Classes

One thing we had to take into account while evaluating was that the COCO dataset's classes do not match up exactly with those of the Failure dataset. There are some classes, such as "barbecue grill", that appear in the Failure dataset but do not appear in COCO. As such, if the YOLO model were to be run on an image containing a barbecue grill, its output would not be useful, since it would be impossible for it to predict the true class. Therefore, for our experiments, we restricted the input to only images containing objects found in both the Failure dataset and the COCO dataset.

3.2. Metrics

Unlike simple classification, there is no single best way to evaluate an object detector. Furthermore, the Failure dataset contains objects that are not annotated because they are not relevant to the failure, as well as objects that are very blurry/unclear and thus highly likely to not be recognized. As a result of this, it is hard to verify whether the algorithm has detected the proper object or not. To combat this, we chose to use a grid search over different IOU thresholds. A threshold of 50%, for instance, would indicate that a prediction must have an intersection over union (IOU) of at least 50% with the ground truth to be considered to have detected the correct object. With this in mind, for each IOU threshold, we were able to calculate:

- Proportion of objects detected: Fraction of objects where at least one prediction met the IOU threshold
- Accuracy: Fraction of detected objects that were classified as the correct class
- Confidence in correct prediction: Average confidence in objects that were correctly classified
- Confidence in incorrect prediction: Average confidence in objects that were incorrectly classified
- Total correct classifications: Fraction of all objects that were classified correctly, regardless of how many were classified incorrectly

With these metrics, we were able to get a fairly clear and complete picture of the strengths and weaknesses of the model.

4. Experiments

4.1. Experiment 1

In the first experiment, we tried IOU threshold values ranging from 0.1 to 0.9, stepping by 0.05. Across these values, proportion of object detections decreased, with a maximum detection rate of 83% at an IOU threshold of 0.1. Accuracy increased from 0.1 to 0.65 (reaching a peak of 92%), then began to drop, before abruptly increasing again at 0.9– the reason for this is unknown, but it may be an outlier due to small sample size, as only 10% of objects were detected at that threshold. These trends were expected – of course more objects will be found at a lower threshold, so the number of detections will be higher. Mistakes where the algorithm classifies a different object that happens to overlap with the ground truth will decrease as the threshold increases, so accuracy should also increase to the point where these mistakes become very unlikely (which experimentally proved to be an IOU of approximately 0.65).

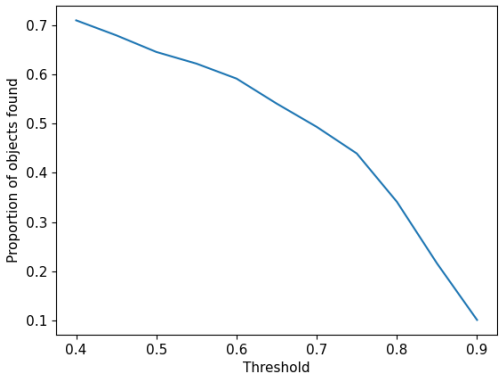

Figure 1. Proportion of objects detected, Exp. 1

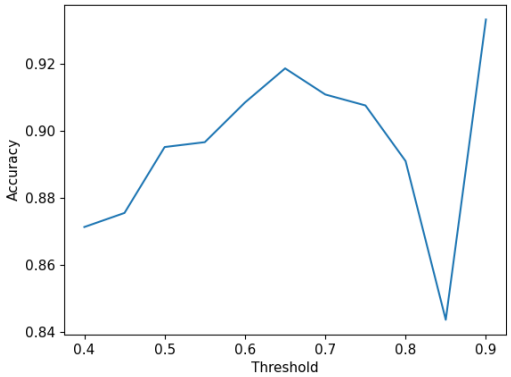

Figure 2. Accuracy, Exp. 1

As the threshold increased, confidence in correct predictions increased, while confidence in incorrect predictions remained roughly the same. This makes sense, as the YOLO algorithm's predictions are more accurate when the bounding boxes match the ground truth more closely, and thus as the threshold increases, the confidence in the correct class should increase. Meanwhile, if the algorithm is predicting the wrong object, then it does not match the ground truth in any event, and it should not matter how much the incorrect bounding box overlaps with the true object's.

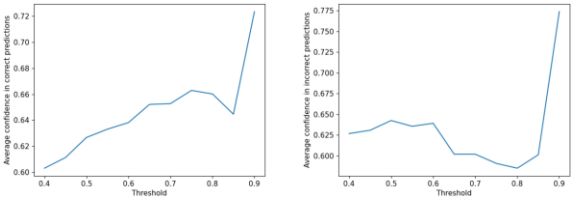

Figure 3. Confidence in correct and incorrect predictions, Exp. 1

Finally, the proportion of objects correctly classified decreased as the threshold increased, from a maximum of 66%. This is the maximum proportion of objects that the algorithm was able to detect at all, if accuracy and proper localization are ignored. This speaks to the difficulty of the dataset; since it does contain many examples of objects that are blurry, obscured, out of focus, or very far from the camera, it is indeed quite difficult to distinguish them.

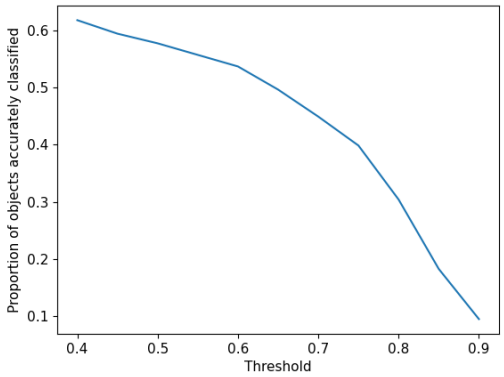

Figure 4. Total correct classifications, Exp. 1

4.2. Experiment 2

In the second experiment, we decided to compare the overall results from both detections. There were 41 target objects in the ground true data. Single-frame detection recognized 32 of them, having a detection rate of 78%, and multi-frame detection recognized 35 of them, having a higher detection rate of 85%. We observed the output images to see the cause of the difference and realized that some target objects could only be detected in a few frames from the videos. For example, the target person (centered one) in Fig 5. was detected in 6 frames out of 10. Single-frame detection suffered a decrease in detection rate when it used the frame in which the target object was unrecognizable. The reason for some objects being unrecognized in some frames is because our input data were frames from videos, in which target objects might be too blurry/unclear to be recognized occasionally. Multi-frame detection prevented from picking only the defective frames, and thus, resulted in a higher detection rate.

In single-frame detection, 25 detected objects were classified correctly, having an accuracy of 78%; in multi-frame detection, 28 detected objects were classified correctly, having a higher accuracy of 80%. The cause of the difference could also be observed in Fig. 5. As we mentioned above, target object might be blurry/unclear in some frames, making it unrecognizable, but it could also cause it to be recognized incorrectly. As you can see in frame 6 and frame 7 in Fig. 5, the target object was misrecognized as a horse and a cow. Single-frame detection received a lower accuracy while taking those frames as the input. On the other hand, multi-frame considered all frames it took, which lower the chance of misclassification.

One more thing we had noticed in experiment two was that there might be more redundant objects detected in multi-frame detection. For example, there was a motorcycle detected in frame 5 in Fig 5. To avoid this, we could apply a threshold value to object appearance times (e.g., keep an object if it appeared in more than 1 frame). By doing this, we could filter out some objects that were not useful for our experiments.

Figure 5. Target object in different frames, Exp. 2

5. Conclusion

Our work is important because it applies a model trained on one dataset to another dataset that is similar while having major differences. The images in the COCO dataset are still images, generally clear and picked to provide good examples of the objects in question, even if the views are not iconic. The Failure dataset, meanwhile, contains real-world examples pulled from the Internet and often recorded on mobile phones (and sometimes edited by apps before being posted). It is important for a model such as YOLO to be able to perform in contexts different from the one it was trained in; learning is being able to generalize from examples to situations that one has never seen, after all.

By setting IOU threshold values properly, we acquire a maximum object detection rate of 83%, a peak accuracy rate of 92%, and a peak combined correct classification rate of 66% on the Failure dataset. Also, we discover that by adding temporal information to the model, we can further improve the performance with 7% increase in the detection rate and 2% increase in the accuracy.

Our contributions were split evenly– Experiment 1 was written and run start-to-finish by Owen, and Experiment 2 was written and run start-to-finish by Michael. We didn't use all videos from the dataset in the second experiment because we couldn't annotate extra 9 frames for all videos in the limited time we had. However, we believe that we would still get an improvement in the performance if we apply multi-frame detection to the full Failure dataset.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Doll´ar. Microsoft coco: Common objects in context, 2015.