Comparison of Object Detection Algorithms: YOLO v3 vs. Single shot Detection

Brandon Gilbert
Department of Computer Science
University of South Carolina Upstate
Spartanburg, South Carolina, USA
gilberb@email.uscupstate.edu

ABSTRACT

Computer vision along with machine learning and artificial intelligence is on the rise and ever evolving. The top auto manufacturers are playing catch up with the market leader Tesla to compete for their market share of consumers interested in a new form of transportation - autonomous vehicles. This has broken new ground in the computer vision scene where new and dedicated stake holders compete using newly develop technologies and take part in the creation of innovative object detection models and algorithms. Research in these fields are the benchmark to the future. The decision of which algorithms can catapult the industry forward will be determined by those collaborators living and breathing in this field.

Keywords

Algorithms, AI, Artificial Intelligence, Machine Learning, Neural Network, Deep Learning, Deep Neural Network, Convolutional Neural Networks

1. INTRODUCTION

Object detection algorithms and computer vision have come a long way since its first development in 1966 by Seymour Papert and Mavin Minsky. This technology led the advancement of the many new and exciting AI based features such as a self-driving cars and various applications used in industrial environments.

Object detection in self-driving cars is an emerging technology that involves meticulous engineering and overwhelming challenges due to risks of life and bodily harm to its occupants. The very nature of roadway deaths stem from some form of human error which result in 93% of all vehicle accidents. These errors are influenced and contributed by many factors of human thought process and mental state. 12 million vehicles were involved in accidents in 2018, just in the United States. Based on human performance, there is one fatality per 100 million miles [3]. This means modern automobiles are vastly safe. Therefore, the error improvement needed by AI is marginally small. This means AI will have to be substantially more efficient at reducing accidents despite the various challenges in today's technology to make a considerable change in these figures. However, by eliminating most human input and having full reliance on a "perfect" artificial intelligence system, this will perceptibly reduce the number of traffic accidents in turn reducing the number of roadway fatalities. This is the goal.

However, owners of these autonomous vehicles will have to put a great deal of trust into the software and code that goes into making these artificial models that allows the automated movement of a 2-ton vehicle in our ever-changing roadways. The AI will be responsible for remaining accurate even in harsh weather conditions and unpredictable situations. Various algorithmic models and deep learning techniques encompass each aspect of self-driving technology. Each of these models contribute to a wide variety of results when tested in a virtual environment and in the field. To effectively determine the usefulness of each model, a series of tests will be completed to determine the model's accuracy, speed, and consistency.

Advancements in the field have greatly improved with the help of academia and technological breakthroughs. Although the essence of a fully autonomous vehicles without human driver assistance may be further along the line until a fully automated vehicle can navigate without error while avoiding obstacles in highly complex roadways and 3D scenes. Until then, self-driving technology will be limited to driver-assistance systems such blind spot monitoring, automatic forward braking assist, and lane detection with intervention. These driver-assist systems are produced and equipped on most new vehicle models today.

However, many technological charged car companies such as EV giant, Tesla, are making huge strides in their fully autonomous systems. This is forcing existing auto manufactures to adopt this approach. Two notable examples are Ford and it's new Co-Pilo360 system and General Motors' Super Cruise system. Currently Telsa's self-driving system only allows a hands-on approach where GM and Ford's systems can be used hands-free legally on over 100,000 miles of highways across United States [6]. This competitive race to make autonomous systems more consumer friendly will ultimately pave way for a more effective and reliable object detection models. In turn, safer roadways, and reduced traffic accidents.

This study will take a small segment from the fully autonomous model and focus on Object Detection and the Algorithms that power them.

This research paper will focus on three key parts to determine the effectiveness of tested models:

- Object detection (number of objects detected)
- Accuracy (in terms of confidence levels (%))
- Speed (processing time to detect object per frame)

2. LITERATURE REVIEW

The current approach in determining an effective model has been related to the speed, consistency, and accuracy produced by the algorithms. The ability the models can effectively predict the objection's location in a given scene and in real-time 3D environments are highly sought after, especially if this prediction process is extremely quick. There are three main tasks that are frequently used to define an object detection model: feature extraction, object classification, and the placement of bounding boxes.

Feature extraction takes the identifiable features of an objection and compares them using a model with certain configurations to a dataset. Then the object classification process starts when the algorithm determines the likely hood that the features detected belongs to an object based on the dataset. Then the classification is made when a certain threshold is met based on the model's configuration settings. During this process bounding boxes are resized and skewed around the predicted object using various resolutions to assist in the objection detection process.

Research on object detection models date back to the 1930s and more heavily in the 1980s with the ALVINN project. This project utilized a neural network that mapped input images to guide and predict steering angles using one hidden layer. After a prolonged AI winter, where research into AI and related topics stagnated, the approach to computer vision research was focused to video games such as it's first implementation in Half -Life and later in Grand Theft Auto V. These tests used highly complexed 3D environments that mimicked real-life roadways combined with simulated physics and traffic control systems. The move to these 3D environments provided researchers a safe and controlled testing environment. These trials were proven successful in using computer vision to test Convolutional Neural Networks to implement basic automated driving in these simulated worlds [3].

More recently, research is conducted on physical test tracks used to evaluate different algorithms that predict pathing. Major breakthroughs in self-driving technology occurred during the 2005 DARPA Grand Challenge. This was fully autonomous race in the desert among five competing universities. This was an amazing feat where Stanford's "Stanley" was the only one of five vehicles to complete the race in record time winning the challenge [5]. Most of the work that derived from that project went into the larger key industry leaders such as Tesla, Google, and Uber which propelled the industry forward. This ultimately paved way for huge leaps in technological advances in autonomous vehicles.

Most current testing methods compare and interpret the results of each model in relation to mean Average Precision(mAP) which is a popular metric that measures the accuracy of objection detectors such as the ones being tested in this study.

The main benefit of this approach is that the accuracy results are produced using 2 boundaries: the ground truth and prediction. This allows the predicts to undergo an IoU threshold which helps identify whether the prediction is a true positive or a false positive. However, this method utilizes a lot more computation resources and is more complex metric. In this study, we will focus our accuracy based on confidence percentages given by the object detection models. This is an quicker method to extract data due to limited nature of this study.

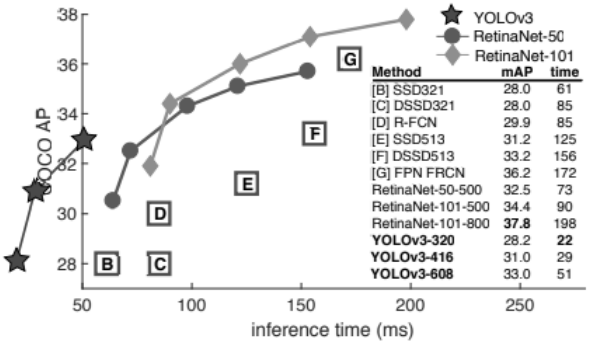

Figure 1. Chart [11]

This chart shows prior research conducted

3. METHODOLODY

The approach in this study is to compare two state-of-the-art single shot object detection models using a robust object classifier model (COCO), weights and config files of each algorithm and commonly used network backbones for each. Data will be obtained during the study and evaluated based on 3 key metrics. These metrics include speed of object detection, accuracy of detection, and overall quantity of objects detected. Ultimately, this section will provide an overview of each algorithm and explain the approach and methods used to compare the two algorithms.

3.1 ALGORITHMS

In the realm of object detection, there are numerous algorithms capable of detecting objects using various media inputs. However, only a select few have the capacity to be both accurate and efficient. Both YOLO v3 and SSD can be easily setup and ran on common hardware. These two algorithms are among the most accurate object detection methods available and very much adequate for our training purposes. The following sections will give a brief overview of each object detection algorithms being compared in this study.

3.1.1 You Only Look Once (YOLO) v3

YOLO is a real-time object detection algorithm which makes its prediction during both training and test-time. YOLO uses a convolutional neural network (CNN) for its real-time object detection and looks at the entire image/frame once before it encodes contextual information to object classification and bounding box determination.

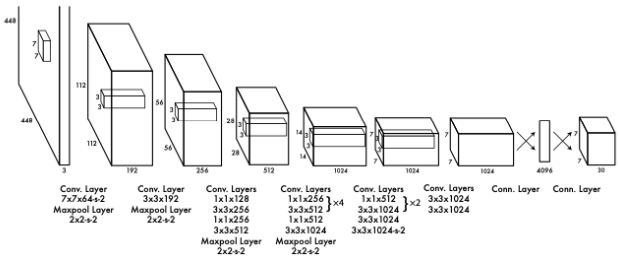

Figure 2. YOLO CNN Architecture [10]

YOLO v3 is the newer, more accurate version from its predecessor YOLO v2, aka YOLO 9000. YOLO 9000 was, at it's time, better in every aspect compared to its predecessor, YOLO v1. YOLO 9000 was the fastest but remained among the most accurate algorithms like RetinaNet and our comparison model in this study, SSD. However, this time YOLO v3 does not retain its speed accolade but in turn trades its speed for boost in accuracy. For example, "the earlier variant ran on 45 FPS on a Titan X, the current version clocks about 30 FPS. This has to do with the increase in complexity of underlying architecture called Darknet [9]."

As mentioned before, YOLO v3 uses CNNs layers, 53 from Darknet-53 and 53 layers from the native architecture. In total, this produces a sum of 106 CNN layers. This is called the Darknet framework and it makes its detection based on 3 layers: 82, 94 and 106. The input image is downsample to the following sizes 32, 16, 8 respectfully to match the detection point layers. This is called Network strides.

Detection layer:	82	94	106
Network strides	32	16	8
Image Size: 416 x 416 (*original input image*)	13x13	26x26	52x52
Detection of object size	Large	Medium	small

Figure 3. Network strides table: This table shows an example of a network input image at 416 x 416 then the network strides will down sample the image at the following network detection layers. In turn, this will determine the size of objects to be detected at each detection layer.

A common issue with previous YOLO versions is that the methods struggle with small objects. This is no longer an issue with YOLO v3 due to its updated darknet-53 network which now utilizes 53 more convolutional layers to perform feature extraction. This was among other updates including a new detection metric, "objectness" scoring for each bounding box using logistic regression, and the use of binary cross-entropy loss for class predictions during trainings.

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2 \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2 \qquad (3)
\end{aligned}
$$

Figure 4. YOLO v2's Loss Function [9]

As shown in Fig. 2, we can see YOLO v2's loss function. For our purposes in this study we will only focus on the last three terms. This is where YOLO v2 differs from YOLO v3 as the terms were squared errors, but in YOLO v3, the object confidence and class predictions are now using cross-entropy error terms based on logistic regression.

3.1.2 SINGLE SHOT DETECTION(SSD)

Much like YOLO, SSD takes only one shot to detect multiple objects within a frame. This is different from other major object detection algorithms such as regional proposal network (RPN) which take two shots. This makes SSD and YOLO much faster.

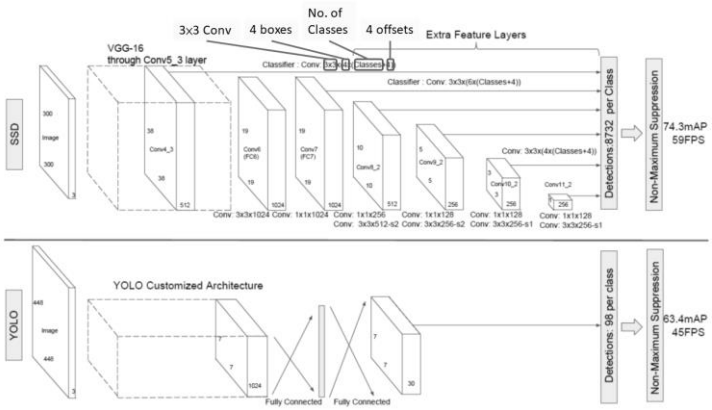

Figure 5. SSD Network Architecture(above), YOLO v1 Architecture (below) [12]

As shown in fig. 4, SSD Network is defined using different layers of feature maps to produce a more accurate detection. During the training phase, SSD only needs the input image and ground truth boxes for each object. Then the input images go through a convolutional series of layers in a small set of boxes in different aspect ratios and scales. This determines the bounding boxes around each object that predicts the shape offsets and confidence levels of object categories. Finally, the boxes are matched with the ground truth boxes and weighed using the localization loss and confidence loss models [13].

3.2 Training Models

To begin the study, models for each algorithm must be trained or utilize a pretrained file object. To expedite this step, I have opted to use pre-trained files for object classification and feature extraction. For YOLO v3 the training model is based on predefined model config and model weight files sourced from the official YOLO v3 website [YOLO]. SDD uses a different feature extractor model called Mobilenetv3. This model uses a frozen inference graph to extract the classification features and produces the maps necessary for object detection. This model is referenced using this file: [frozen_inference_graph.pb] which contains both weights and configurations for use on the COCO data set and Mobilenet.

3.3 Bounding Box Encoding

Both algorithms contain similar steps in this section. I will attempt to cover the differences and explain similarities between the two.

3.3.1 YOLO v3

During training YOLO v3 applies CNN to image that down samples image at 3 scales. In turn, this creates 1x1 kernels that are applied to a grid of cells.

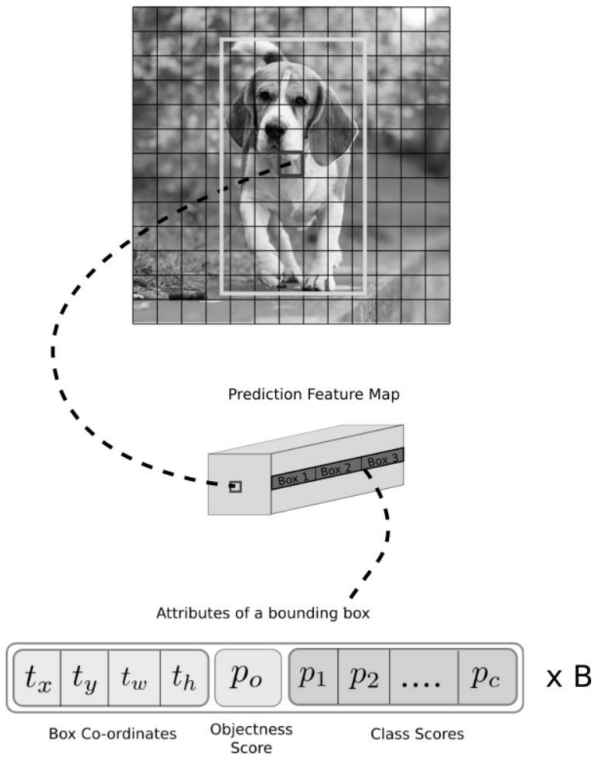

Image credits: https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

Figure 6. Prediction of Feature Map 1x1 kernel

Attributes of bounding box includes Box dimensions, Objectness Score and Class Scores [9]

For YOLO v3, the shape of detection kernels is determined by the following equation:

$$(b * (5 + c)),$$

Where:

b = 3, number of bounding boxes (BB). This number is based on the number of feature maps. In our case there are three based on Fig. 3. (13x13), (26x26), (52,52)

Each bounding box has (5 + c), BB attributes

c = 80, COCO classes (this is the dataset used in this study)

Result:

$$(3 * (5 + 80)) = \mathbf{255\ attributes}$$

Detections Kernels' feature maps would be the following:

$$(13,13,255)$$
$$(26,26,255)$$
$$(52,52,255)$$

This means that each cell contains 3 bounding boxes and it predicts an object through one of its bounding boxes if the center of the object fits inside the corresponding box.

The bounding boxes size is determined based off anchor boxes (predefined BB). In YOLO v3, there are a total of 9. That means there are 3 for each scale. We will not go over the equations of determining the box size, but I will explain that YOLO v3 has over 10x the number of predicted boxes compared to YOLO v2. In our running example of our input image of 416 x 416, this would be 13 x 13 cells across 3 bounding boxes and 80 classes' confidences at three scale levels: Scale - 1: 507, Scale - 2: 2028. Scale - 3: 8112. This would create a prediction of 10,647 bounding boxes [9]. These are then filtered to only allow the correct ones.

The Objectness score predicts the probability that bounding box contains the object inside. It uses the following equation:

$$\boldsymbol{P}object * \boldsymbol{IoU} = \boldsymbol{\sigma}(t_0) = P_0$$

Pobject – predicted probability that BB contains object

IoU – Intersection of Union between predicted BB2 and ground truth BB1

Class Scores are the confidence scores that determine if the object belong to certain class of object (person, cat, car, etc.)

3.3.2 SSD

In the SSD Architecture, a network model is used to extract feature map. In this study, we will be using MobileNet. Then the image goes through 6 CNN layers that make a total of 8732 bounding boxes for the object detection. Finally, the predicted BB are filtered by Non-Max Suppression. The task of the CNN Layer is to create various default boxes of different sizes and aspect ratios across the entire image.

4. IMPLEMENTATION

Ideally, the implementation of this study would take place in a live environment using a physical platform such as a drone, or robot. However, due to time restraints, the study will be done completely on a virtual machine using Linux on static images and video. Although, this study will have a brief comparison of both algorithms using live webcam feeds. This will be discussed in more detail in the Experiment Setup section. This section will break down all the individual components used to create the program running this comparison and offer key details on the programming language, system, and dependencies used.

4.1 Programing Environment

This study will be coded using Python 3.8 on an Ubuntu Linux virtual machine. This allows seamless setup on a native environment and allows the user the ability to download and install Python and project dependencies with ease. The dependencies used to support the programs running the compares and training models are OpenCV(cv2) and NumPy. OpenCV is an open source computer vision library that contains various functions that can be used with the Python language [OpenCV].

NumPy is also an open source project but aims to enable greater numerical computing within the Python programming environment. NumPy contains various high-level mathematical functions to support large multi-dimensional arrays and matrices [**NumPy**]. Another library used in this implementation is Matplotlib. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python[Matplotlib].

5. EXPERIMENT SETUP

In this section, the experimental setup of the comparison between YOLO v3 and Single Shot Detection will be explained. The data set of the study will be described and given reason to why it was chosen. Each evaluation metric will also be explained in detail.

5.1 Dataset

For this study, it will be implemented using the Common Objects in COntext (COCO) dataset. COCO is a powerful dataset that can classify an object using 80 different object categories among 91 stuff subcategories. This dataset is made up of 330K images with over 1.5 million object instances and as the ability to classify each object with up to 5 captions. The COCO dataset utilizes object segmentation that allows objects to be separated within a given frame despite clutter, partially occluded scene objects, and complex backgrounds.

Although there are many datasets that achieve similar goals and some with even more object categories such as Sun (over 6,000) and ImageNet (Over 10,000), however their object instances for each category and image is relatively low compared to COCO. When deciding which dataset to use, we often consider the reliability of the data and how much data is available. In relation to COCO's 80 object categories compared to ImageNet's tens of thousands, we must look at how that data will be used. In this study, the determination was to consider the use-case. For this study, the use-case to learn about two complex object detection algorithms and evaluate each of them based on a series of metrics. Ultimately, the data must be stable and consistent to retrieve usable results from the evaluation metrics. Since COCO has more object instances, this is a more reliable dataset for training and testing purposes. To back up this claim, COCO's dataset contains 3.5 categories and 7.7 instances per image whereas ImageNet has an average of less than 2 categories and 3 instances per image [6].

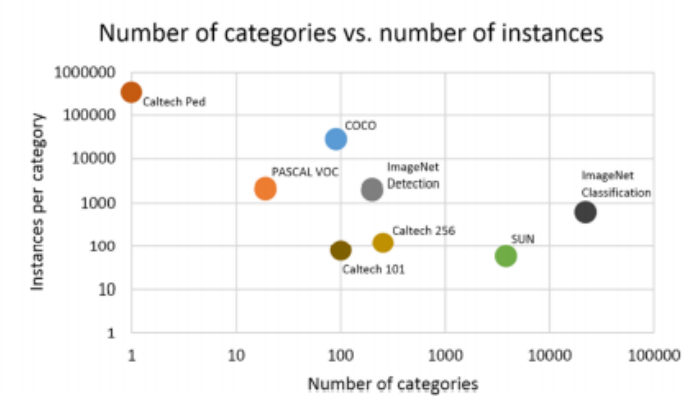

Figure 7. Number of categories vs. number of instances [6]

As shown in Figure 1, we can determine COCO has one of the highest ratios of Instances over categories among the top object classification models. This finally persuaded the use-case of COCO in this study.

In addition, COCO also has many user-friendly features that allow the dataset to actively evaluate itself using a built in Python script, evalCapDemo. This script will be used to assist in the evaluation metrics in this study.

5.2 Evaluation Metrics

5.2.1 Speed and Number of Objects detected

When determining the effectiveness of a model, one of the most important metrics is speed. The speed at which the model can detect an object and determine its classification is vital to any test. This calculation will be based on the number of objects detected divided by the length of video in milliseconds (ms). During live webcam feed, the determination of this metric will be based on the length of live feed also in milliseconds.

This metric will be calculated using the process time it takes each algorithm to complete its objection detection classification over the course of test video.

The number of objects detected will be calculated by summing up the number of objects detected per algorithm.

5.2.2 Accuracy

One of the most important pieces of data to extract from this study is accuracy of each model. Based on the concept of object detection, each algorithm makes it object determination based on a series of computations such as confidence levels, bounding boxes comparisons, and loss functions. Due to this very nature, this study will use the bounding boxes confidence levels as a correlation to accuracy.

Although these confidence levels may have no relation to the exactness of object being determined as object may be misrepresented, but this will give a good idea of which algorithm has the most confidence in object classification when tested. In theory, the higher the confidence level usually correlations to an accurate depiction of the object.

5.2.3 Consistency

The consistency of the study is vital. This means the program(s) in this study will be ran a multiple of times (Min. of 3). The average of the Speed, Number of objects detected, and Accuracy will be averaged. The average will be the base scoring metric for use during the results analysis and conclusion determination in sections 6 and 7.

5.3 Experimental Setup

The goal of this study is to compare YOLO v3 to SDD and determine which algorithm is better using the metrics outlined in section 5.2. To start, the experiment will be setup to have a Python program using pre-defined training models on the selected network backbones. Then a series of tests will be ran using the same video and live-webcam feed. These tests will run a minimum of three times each to meet the consistency metric. Then the evaluation metrics will be extracted using built-in Python scripts with the library dependencies Matplotlib and evalCapDemo to create graphs and data tables. The data will then be analyzed, and a conclusion will be made.

6. RESULTS ANALYSIS

This section's purpose is to analysis the results of the study conducted between the two object detection algorithms. During this study, four different tests were ran to determine the best possible object detection model to include Number of Objects Detected, the Confidence Levels Per Frame, The Process Time per Algorithm, and finally the Number of Objects Detected by Class ID.

All tables shown in this section were calculated using the data from the experiment and the results are displayed using Matplotlib. For Figure 11., this data was extracted using Pandas and stored into an .csv file format and is currently shown as an Excel Table for this study.

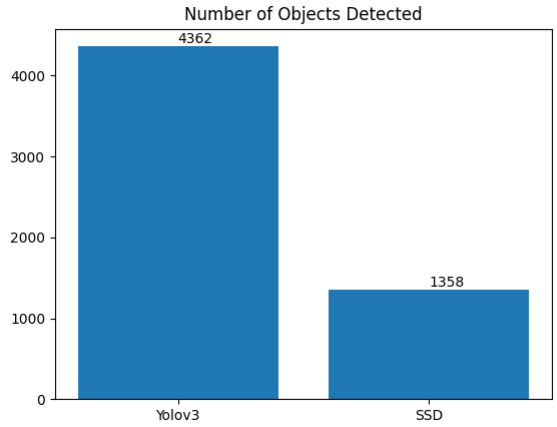

Figure 8. Number of Objects Detected

This bar graphs displays the number of total objects detected by each algorithm. As shown, Yolov3 detected an outstanding 4x the number of objects compared to SSD's 1358 during our testing. This is a substantial number of objects compared to SSD, thus ruling Yolov3 the victor in this test.

However, this test does not determine whether the objects detected were accurate and does not remove false positives. This test only counts the number of bounding boxes created with the associated classification identified by each algorithm.

Figure 9. Confidence Levels Per Frame

This line chart displays the average confidence levels per frame for each object detection algorithm. The higher the confidence levels the better. During this test, it was determined that Yolov3 stood out with an average of 30% greater confidence levels per frame compared to SSD. Additionally, Yolov3 held these levels with great support without any percentages outside its normal range. SSD, on the other hand, had periods were the algorithm struggled to hold its confidence levels around 65% and have a few frames where the percentages dropped to zero.

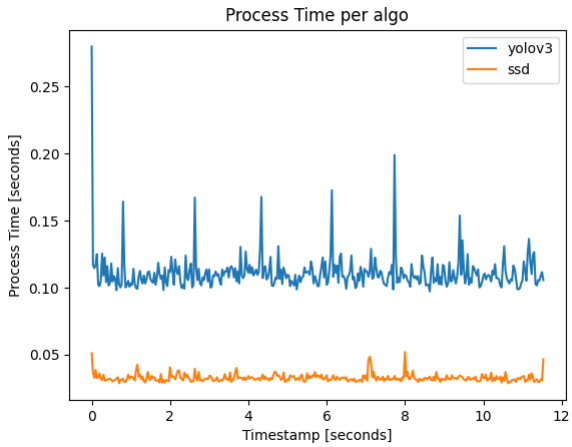

Figure 10. Process Time per Algorithm

Now this line chart changes the tide of this particular test by placing SSD in number 1 position. This chart shows the time each algorithm takes to process each objection detection calculation to include building the bounding boxes. The lower the process time, the better for this test. As we can see, SDD stands out with much quicker processing times. This can due to a variety of reason such as the complex nature of the algorithm itself or the environment the algorithm was running in.

One thing to take note is that both SSD and Yolov3 were ran using the same environment listed in the Section 4.1. However, Yolov3 run more efficiently using a NVidia GPU compared to the CPU used in this study. Results may be different if ran using a GPU, but this was not conducted in this study.

Number of Obects Detected by Object ID		
Object IDs	yolov3	SSD
person	1480	527
bicycle	6	0
car	2726	687
motorbike	109	0
traffic light	39	137
tvmonitor	2	0
bus	0	7

Figure 11. Number of Objected Detected by Object ID

This table shows the number of objects detected by Object ID. These number are shown as objects detected per frame; thus the

numbers are much higher than what is shown in the video. As we can see, Yolov3 detected over 3 times the number of objects and detected some objects SDD was not able to such as bicycles and motorbikes.

7. CONCLUSION

In conclusion, we have conducted multiple tests to determined whether Yolov3 or SDD is the better object detection algorithm. We have ran the tests in a neutral environment to offer and receive unbiased results. These tests were also conducted multiple times to ensure consistency. Both algorithms have their own use cases and strengths. However, in this study we have determined that **Yolov3 is the better algorithm compared to SSD** when comparing the results from Section 6.

When analyzing the results using the figures provided in Section 6, it is clear that Yolov3 outperformed against SSD. Yolov3 performed over 30% better in 3 out of 4 tests. In comparison, SSD only outperformed against Yolov3 when comparing processing times. As I mentioned before, the processing times may be different if a GPU were to be used during this test. When determining an object detection algorithm for real-life use cases such as self-driving cars and industrial processes, we must choose the object detection model that is more accurate and reliable. In this case, Yolov3 exceled in this study when in comparison to SSD. However, at the time of writing this report, both of these object detection algorithms are obsolete with newer versions already available. To objectively choose an algorithm to best fit a real-life task or purpose, more research will be needed to compare these algorithms in this study to the newer released versions.

8. REFERENCES

[1] M. Daily, S. Medasani, R. Behringer and M. Trivedi, *"Self-Driving Cars,"* in Computer, vol. 50, no. 12, pp. 18-23, December 2017, doi: 10.1109/MC.2017.4451204.

[2] Simhambhatla, Ramesh; Okiah, Kevin; Kuchkula, Shravan; and Slater, Robert (2019) "Self-Driving Cars: Evaluation of Deep Learning Techniques for Object Detection in Different Driving Conditions," *SMU Data Science Review*: Vol. 2 : No. 1 , Article 23..

[3] Martinez, M., Sitawarin, C., Finch, K., Meincke, L., Yablonski, A., and Kornhauser, A., "*Beyond Grand Theft Auto V for Training*, Testing and Enhancing Deep Learning in Self Driving Cars", arXiv e-prints, 2017.

[4] Ni J, Chen Y, Chen Y, Zhu J, Ali D, Cao W. *A Survey on Theories and Applications for Self-Driving Cars Based on Deep Learning Methods*. Applied Sciences. 2020; 10(8):2749. https://doi.org/10.3390/app10082749

[5] Thrun S. (2006) *Winning the DARPA Grand Challenge*. In: Fürnkranz J., Scheffer T., Spiliopoulou M. (eds) Machine Learning: ECML 2006. ECML 2006. Lecture Notes in Computer Science, vol 4212. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11871842_4

[6] Lin TY. et al. (2014) *Microsoft COCO: Common Objects in Context. In: Fleet D*., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_48

[7] Gilboy, J. (2020, June 18). *Ford's Answer to Autopilot and Super Cruise Is Co-Pilot360 on the Mustang Mach-E*. The Drive. https://www.thedrive.com/tech/34184/fords-answer-to-autopilot-and-super-cruise-is-co-pilot360-on-the-mustang-mach-e.

[8] Wagner, I. (n.d.). *Topic: Road accidents in the United States*. Statista. https://www.statista.com/topics/3708/road-accidents-in-the-us/.

[9] Kathuria, A. (2018, April 29). *What's new in YOLO v3?* Medium. https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b.

[10] *YOLO - object detection*¶. YOLO - object detection - OpenCV tutorial 2019 documentation. (n.d.). https://opencv-tutorial.readthedocs.io/en/latest/yolo/yolo.html.

[11] J. Redmon and A. Farhadi. Yolov3: An incremental improve ment. arXiv, 2018. 4

[12] Tsang, S.-H. (2019, November 11). *Review: SSD - single SHOT Detector (OBJECT DETECTION)*. https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11.

[13] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., & Berg, A. (2016). SSD: Single Shot MultiBox Detector. *ECCV*.