ARL-TR-9397 ● FEB 2022

# Baseline Assessment of Object Detection Models on Partially Occluded Objects

by Darius Jefferson II

## NOTICES

### Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9397 ● Feb 2022

# Baseline Assessment of Object Detection Models on Partially Occluded Objects

**Darius Jefferson II**
*Computational and Information Sciences Directorate,*
*DEVCOM Army Research Laboratory*

| REPORT DOCUMENTATION PAGE | *Form Approved*<br>*OMB No. 0704-0188* |
|---|---|

| | | |
|---|---|---|
| **1. REPORT DATE** *(DD-MM-YYYY)*<br>February 2022 | **2. REPORT TYPE**<br>Technical Report | **3. DATES COVERED (From - To)**<br>17 June 2020–30 September 2021 |

| | |
|---|---|
| **4. TITLE AND SUBTITLE**<br>Baseline Assessment of Object Detection Models on Partially Occluded Objects | **5a. CONTRACT NUMBER** |
| | **5b. GRANT NUMBER** |
| | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)**<br>Darius Jefferson II | **5d. PROJECT NUMBER** |
| | **5e. TASK NUMBER** |
| | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**<br>DEVCOM Army Research Laboratory<br>ATTN: FCDD-RLC-IB<br>Adelphi, MD 20783-1138 | **8. PERFORMING ORGANIZATION REPORT NUMBER**<br>ARL-TR-9397 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** | **10. SPONSOR/MONITOR'S ACRONYM(S)** |
| | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

Approved for public release: distribution unlimited.

**13. SUPPLEMENTARY NOTES**

ORCID ID: Darius Jefferson II; 0000-0002-4538-084X

**14. ABSTRACT**

One of the fields of computer vision commonly used in military research is object-detection. A particularly good example of this is real-time object recognition on the battlefield. Developing/evaluating these types of models requires proper object-detection and classification datasets, which are crucial for Soldiers' decision-making on the battlefield. A major problem with current object-detection models is that they flounder when detecting partially occluded objects. This is because the models do not properly recognize the objects while parts of them are covered. Additionally, occlusion is not a condition that many object-detection models are designed to handle. The main objective of this work was to perform a baseline assessment of the Gonzalez–Garcia model compared with the Faster R-CNN model from Detectron2 and YOLOv5 using the PASCAL VOC 2010 dataset. Of course, this dataset contains many examples of partially occluded objects. The results from each would then be compared to determine their overall effectiveness and their accuracy on partially occluded objects. All three object-detection models seem to work well overall and somewhat well with partially occluded objects. However, none of them are very good at detecting objects in poor lighting conditions.

**15. SUBJECT TERMS**

object-detection, occlusion, parts detection, VOC, computer vision, YOLO, Detectron

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON**<br>Darius Jefferson II |
|---|---|---|---|---|---|
| **a. REPORT**<br>Unclassified | **b. ABSTRACT**<br>Unclassified | **c. THIS PAGE**<br>Unclassified | UU | 24 | **19b. TELEPHONE NUMBER (Include area code)**<br>(301) 394-1404 |

## Contents

## List of Figures

## List of Tables

## 1. Introduction

Object detection is one of the most popular fields of computer vision used for military applications. One of the ways in which object-detection models are used in this context is for real-time object recognition on the battlefield. Many of these models are starting to be incorporated into technology used by Soldiers (i.e., unmanned ground vehicles and heads-up displays) to assist them in identifying objects around them that could represent potential threats to their safety. By properly detecting and classifying hazardous objects on the battlefield, the models could be able to provide Soldiers with useful information about their surroundings so they can make decisions regarding how to proceed in their missions.

A major problem that occurs with current object-detection models is that they have trouble detecting objects that are only partially visible or occluded. In these cases, object-detection models will often miss detecting these objects at all. They may also detect the partially occluded objects but then classify them using the wrong object class. Occlusion is a condition that many researchers do not account for when developing and training their object-detection models even though it is commonly seen in the real world. To ensure the safety of Soldiers, as well as improving the state of object-detection models in the future, it is necessary to determine how well current object-detection models work when faced with this scenario.

The main objective of this work was to perform a baseline assessment on three state-of-the-art object-detection models on a popular object recognition dataset containing many partially occluded objects. After doing so, the results from each were compared. The models used in this experiment are the Gonzalez–Garcia model,[1] Faster R-CNN from Detectron,[2] and YOLOv5.[3] The dataset in which they were trained and tested on was one of the popular Pattern Analysis, Statistical Modeling and Computational Learning Visual Object Classes (PASCAL VOC) challenge datasets, specifically VOC 2010.[4] This report begins by presenting an overview describing each of the object-detection models and the VOC dataset. Then more details about the experiment are given, along with results and the conclusion.

## 2. Gonzalez–Garcia Model

The Gonzalez–Garcia model (GG) is a MATLAB-based object-detection model. Like a typical object-detection model, the GG model detects the whole object but also the semantic parts of that object.[1] "Semantic" parts are those sections of an object that can be easily recognized and described by humans.[5] This subset field of object detection is known as parts-of-object-detection or parts detection. The reason it does this is to improve object detection by being able to identify objects by their parts, the combination of which are often unique to each class of object.

This model uses a convolutional neural network (CNN) as a backbone, specifically Fast R-CNN.[6] As a result, images used as input are sent through multiple convolutional layers so that region-of-interest (ROI) pooling may be performed. ROI pooling creates what are known as region proposals from the input images; various areas from within the image considered noticeable by the model. Each image contains two types of region proposals, one for objects and one for parts. It is at this point that the model splits into four separate branches.

Part proposals are sent to both the part appearance and relative location branches, both of which contain two fully connected (FC) layers. The part appearance branch focuses on classifying parts based on their appearance, as the name implies. The relative location branch uses the location of parts relative to their respective objects as a way of identifying those parts. For example, if an identified object within an input image is a car, the relative location branch could identify areas underneath the car as containing wheels. It would then score the part proposals based on their overlap with the areas suggested by the relative location branch. These “suggestions” made by the relative location branch are done using a separate pretrained CNN that the model’s researchers created for this purpose called OffsetNet.[1]

Object proposals are sent to both the object class and object appearance branches. The object class branch tries to correctly classify objects found within the proposal. As mentioned, knowing the class of an object is not only important for the model’s effectiveness overall, but also useful for the relative location branch for identifying an object’s semantic parts. The object appearance branch, like the part appearance branch, identifies objects based on their appearance. The object class branch contains three FC layers while the object appearance branch contains only two.

The outputs from the part appearance branch and from the two object branches are concatenated to form one unified part representation, which is then scored and fed into a regressed bounding box layer. At the same time, the relative location branch computes its own scores for parts. This is done separately since not all parts benefit from knowing their relative location within an object. Afterward, the scores from the unified part representation and the relative location branch are linearly combined to form the final outputs of the GG model.

The initial results from the developer’s paper show that the GG model works considerably better than models that focus on part appearance only to identify parts, including on partially occluded objects.

## 3. Detectron2

Another state-of-the-art model investigated and then used for this research was Faster R-CNN. However, the original implementation of this model had already been deprecated by the time this project began. One of the most recent implementations of it, and one suggested by the original developers of the model, was contained within Detectron2.

Detectron2 is an object-detection platform developed by the Facebook Artificial Intelligence Research team and released in 2018.[2] As the name implies, it is the successor to the original Detectron, which functioned similarly and was developed by the same team. As an object-detection framework, the purpose of Detectron2 is to provide modern and high-quality implementations of various object-detection models through their model zoo. These include, of course, Faster R-CNN,[7] Mask R-CNN,[8] RetinaNet,[9] DensePose,[10] Cascade R-CNN,[11] Panoptic FPN,[12] and TensorMask.[13]

All model implementations are written in Python and use the PyTorch Deep learning library. They were also designed with the capability of being used with either single or multiple graphics processing units (GPUs) for training.

## 4. YOLOv5

The last of the three models looked at was YOLOv5, the fifth version of the YOLO series of object-detection models.[3] YOLO, which stands for "You Only Look Once", was originally designed to perform quick object-detection by applying entire input images to be analyzed by a neural network.[14] The neural network would then divide each image into regions to be predicted upon and weighted. Since the model only evaluates using the neural network once per image as opposed to the hundreds or thousands of times used by other models, it makes using YOLO much faster in comparison.

YOLOv5 is the latest successor to YOLOv3, which was the last version of YOLO developed by its original developers.[15] YOLOv5 was developed at the same time as another YOLOv3 successor (developed by a different group of researchers) known as YOLOv4 and was released only a month after it. YOLOv5 offers several GPU architecture variations, including some that were pretrained on the Common Objects in Context (COCO) dataset[3]. It was developed by the company Ultralytics in May 2020, and the model incorporates the knowledge they obtained from numerous hours of research into future vision AI methods. YOLOv5 is written in Python and uses PyTorch for deep learning.

The results from the developer's GitHub page show the accuracy for each variation of YOLOv5 when trained and validated on the COCO dataset. It clearly demonstrates the improved average precision and GPU speed of each architecture variation as you increase the size.

## 5. PASCAL VOC

PASCAL VOC 2010 is the name of the dataset used for training and testing the models in this report.[4] PASCAL VOCs are ever-expanding image datasets standardized for object class recognition. These datasets were originally made for the PASCAL VOC series of challenges that spanned 2005–2012, with each year having its own datasets.[16] One of the main reasons that the VOC 2010 dataset was chosen was because this was the original dataset used to train the GG model, which was the first model investigated. The other reason is that the GG model's developers had created a semantic parts dataset using VOC 2010 that will be used in this research in the future.[1]

Every VOC dataset since 2007 has two things in common. The first is that all objects can be classified into at least one of 20 object classes, which feature mainly different animals and vehicle types. The second is that all image annotations are done using XML files, which contain a variety of tags describing the objects within the image.

Two of the most important XML tags considered in this report are the "occluded" and "difficult" tags. The "occluded" tag contains a binary value indicating whether an object within an image is occluded (a "1" means it is occluded). The "difficult" tag contains a binary value denoting whether an object is considered difficult to detect (a "1" means it is difficult). Often, this "difficulty" can be attributed to poor lighting conditions within the image. Note in the original PASCAL VOC challenges, any objects marked as difficult were skipped in the evaluation process. Examples of objects marked occluded or difficult can be seen in Fig. 1.

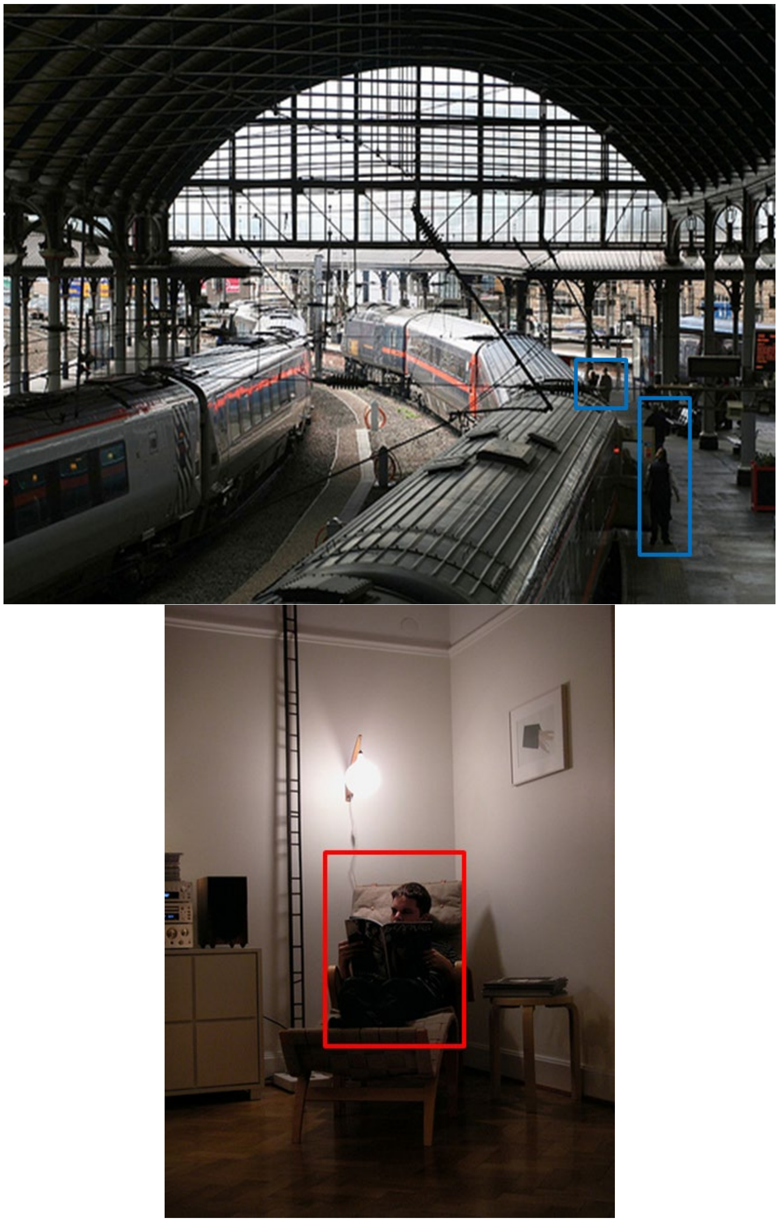

**Fig. 1 (top) A VOC 2010[4] image containing "difficult" objects (the people, highlighted in blue); (bottom) a VOC 2010[4] image containing an "occluded" object (the person, highlighted in red)**

## 6. Experiment

The crux of this research was to be able to determine the effectiveness of each of the three object-detection models on the PASCAL VOC 2010 dataset. Each model would be trained on 2010's training set and tested/evaluated on its validation set since the testing set was not publicly available. The results of each were quantified by measuring the average precision (AP) for each object class. AP is the measure of the model's precision versus its recall and is a popular metric used for determining a model’s accuracy. Once you have the APs from each object class, you can find the mean average precision, or mAP, of the entire model. mAP is the mean of all the APs across all of the object classes.

The object class APs were found based on eight cases/categories. The aggregate case was based on the rules of the VOC challenges, which excluded difficult objects from the evaluation. This case is also the one used to find each model’s mAP. The remaining seven categories were referred to as the breakout categories and described the conditions the evaluated objects were under. These categories included occluded, unoccluded, difficult, occluded and difficult, occluded and non-difficult, unoccluded and difficult, and unoccluded and non-difficult.

These categories are mostly combinations of the two XML tags that were mentioned before, both of which are very important when determining effectiveness on partial occlusion. When finding the class AP, any objects whose tags do not match the breakout category were not counted among the true positives.

## 7. Results

The experimental results are presented in Table 1.

**Table 1 Aggregate object class AP results from the GG model, Detectron2, and YOLOv5**

| Object class APs (all models) | | | |
|---|---|---|---|
| **Object class** | **GG** | **Detectron2** | **YOLOv5** |
| aeroplane | 75.37% | 82.25% | 88.10% |
| bicycle | 68.85% | 56.69% | 86.60% |
| bird | 56.80% | 61.64% | 81.60% |
| boat | 36.08% | 56.86% | 69.80% |
| bottle | 26.86% | 64.59% | 73.20% |
| bus | 71.26% | 78.21% | 88.60% |
| car | 58.48% | 84.08% | 85.50% |
| cat | 77.61% | 70.84% | 87.60% |
| chair | 23.58% | 51.21% | 65.30% |
| cow | 45.54% | 64.75% | 74.20% |
| dining table | 35.99% | 51.63% | 57.80% |
| dog | 72.21% | 62.20% | 83.90% |
| horse | 65.33% | 83.28% | 85.70% |
| motorbike | 70.96% | 87.07% | 87.80% |
| person | 63.22% | 87.45% | 86.70% |
| potted plant | 25.45% | 54.83% | 57.90% |
| sheep | 60.14% | 75.40% | 82.00% |
| sofa | 37.85% | 59.33% | 67.40% |
| train | 72.19% | 79.92% | 87.70% |
| tv monitor | 58.01% | 58.97% | 79.80% |

Based on the current results, it is clear that YOLOv5 is the best model of the three for general object-detection purposes. Its aggregate mAP is the highest at 78.90%, while Faster R-CNN is 68.56%% and GG is at 55.09%, which can be seen in Table 2. Incredibly, YOLOv5 has over 80% AP in 12 out of the 20 total classes. The next closest would be Detectron2, but it only has five while GG has none.

**Table 2 Aggregate object-class mAPs from the GG model, Detectron2, and YOLOv5**

| **Aggregate object mAPs (all models)** | |
|---|---|
| GG | 55.09% |
| Detectron2 | 68.56% |
| YOLOv5 | 78.90% |

All three models share some commonalities in their aggregate object-class APs. The object class that contains the highest AP is not the same between each of them. However, as seen in Table 1, the set of classes with the highest APs are about the same among them. This includes aeroplane, car, cat, horse, motorbike, and person. The most likely reasons for this is either because they have lots of examples in the training set (car, cat, person) or have very distinctive sizes/shapes compared with other classes (aeroplane, horse, motorbike).

The set of classes with the lowest aggregate APs are also about the same between each of them, which include the chair, dining table, and potted plant classes. In the case of the dining table class, the APs are probably low due to the amount of training examples being lower than many other classes. For chair and potted plants, each of those have various types just like the car class. Unlike the car class, which mostly maintains the same overall shapes, chair and potted plant classes can be more varied. Perhaps this makes it more difficult for the model to correctly detect those classes.

For the occluded category, all models shared similar classes for the highest four class APs. In GG's case, the highest four APs were in the motorbike, horse, bicycle, and person classes (from Table 3). In Detectron2's case, the highest four APs were the person, horse, motorbike, and car classes (from Table 4). In YOLOv5's case, the highest four APs were dining table, bicycle, horse classes, and person classes (from Table 5). Although Detectron2 and YOLOv5 do not perform any form of explicit parts detection natively, it could still be argued that the high APs in these classes may be because of how recognizable their parts are even under occlusion. The worst occluded AP between both GG and Detectron2 was the boat class, while the worst for YOLOv5 was the aeroplane class.

As seen in Tables 3 and 4, the highest AP in the difficult category for all models was the dining table class. Presumably, this is because while considered difficult to detect, a dining table is still very large and hard to mistake for anything else when not occluded. All other APs for all models are relatively low (with the exception of the person class in Detectron2's case), presumably due to the difficulty.

**Table 3 GG model's object class APs for the occluded, unoccluded, and difficult breakout categories**

| **GG detected objects (occluded/unoccluded/difficult) Part 1** | | | |
|---|---|---|---|
| **Object class** | **Occluded** | **Unoccluded** | **Difficult** |
| aeroplane | 18.60% | 74.12% | 0.13% |
| bicycle | 52.57% | 68.24% | 0.18% |
| bird | 22.29% | 54.39% | 0.02% |
| boat | 3.39% | 38.20% | 0.82% |
| bottle | 5.45% | 28.47% | 0.42% |
| bus | 33.17% | 75.31% | 0.26% |
| car | 25.89% | 62.48% | 0.28% |
| cat | 43.07% | 80.96% | 1.18% |
| chair | 8.70% | 23.18% | 0.62% |
| cow | 23.63% | 40.96% | 1.13% |
| dining table | 30.55% | 27.53% | 14.06% |
| dog | 38.83% | 74.20% | 0.05% |
| horse | 49.46% | 65.10% | 0.37% |
| motorbike | 55.49% | 68.28% | 1.15% |
| person | 47.50% | 56.57% | 2.22% |
| potted plant | 7.28% | 25.42% | 1.03% |
| sheep | 30.32% | 58.44% | 0.86% |
| sofa | 24.51% | 26.42% | 2.67% |
| train | 46.56% | 74.56% | 0.51% |
| tv monitor | 16.77% | 62.86% | 0.03% |

**Table 4 Detectron2's object class APs for the occluded, unoccluded, and difficult breakout categories**

| **Detectron2 Faster R-CNN detected objects (occluded/unoccluded/difficult) Part 1** | | | |
|---|---|---|---|
| **Object class** | **Occluded** | **Unoccluded** | **Difficult** |
| aeroplane | 29.24% | 84.54% | 8.64% |
| bicycle | 46.81% | 56.51% | 4.71% |
| bird | 29.40% | 60.16% | 0.00% |
| boat | 11.47% | 63.75% | 7.24% |
| bottle | 45.64% | 59.43% | 1.94% |
| bus | 52.32% | 79.65% | 1.75% |
| car | 65.47% | 81.40% | 8.73% |
| cat | 52.63% | 74.27% | 0.93% |
| chair | 31.08% | 53.56% | 3.36% |
| cow | 38.27% | 67.47% | 8.21% |
| dining table | 50.90% | 45.64% | 43.13% |
| dog | 38.93% | 63.82% | 0.00% |
| horse | 75.71% | 80.76% | 0.87% |
| motorbike | 75.19% | 85.09% | 7.68% |
| person | 76.79% | 83.69% | 21.47% |
| potted plant | 25.35% | 54.52% | 1.61% |
| sheep | 53.56% | 71.63% | 4.91% |
| sofa | 43.97% | 53.99% | 8.26% |
| train | 62.77% | 82.76% | 1.43% |
| tv monitor | 28.26% | 65.26% | 3.33% |

**Table 5 YOLOv5's object class APs for the occluded, unoccluded, and difficult breakout categories**

| **YOLOv5 detected objects (occluded/unoccluded/difficult) Part 1** | | | |
|---|---|---|---|
| **Object class** | **Occluded** | **Unoccluded** | **Difficult** |
| aeroplane | 5.36% | 86.10% | 1.03% |
| bicycle | 39.50% | 53.90% | 0.45% |
| bird | 10.50% | 71.60% | 0.13% |
| boat | 7.11% | 68.80% | 2.78% |
| bottle | 16.60% | 58.10% | 0.68% |
| bus | 16.10% | 81.90% | 0.45% |
| car | 25.90% | 68.00% | 2.14% |
| cat | 13.40% | 79.50% | 0.45% |
| chair | 25.20% | 46.00% | 2.57% |
| cow | 20.80% | 58.40% | 0.93% |
| dining table | 42.50% | 27.50% | 9.38% |
| dog | 14.00% | 75.50% | 0.09% |
| horse | 39.20% | 54.10% | 0.00% |
| motorbike | 33.80% | 60.60% | 0.67% |
| person | 38.70% | 49.00% | 2.53% |
| potted plant | 14.10% | 46.90% | 0.83% |
| sheep | 23.20% | 62.40% | 1.65% |
| sofa | 36.90% | 39.30% | 3.23% |
| train | 17.40% | 78.00% | 0.28% |
| tv monitor | 10.90% | 72.50% | 0.67% |

The last breakout category of particular interest is also the hardest, the occluded and difficult case. Once again, the highest AP for all models was the dining table class (from Tables 6–8). While it is natural to think that this category would be equally influenced by class AP results from both the occluded and difficult categories, this does not always seem to be the case. In the cases of the GG and Detectron2 models (as seen in Tables 6 and 7), the results show that it seems to be mostly dependent on how well the models did in the corresponding difficult classes. Especially since the dining table class is relatively average for both models in the occluded category, but only high in the difficult category. Similar patterns can be observed in their other class APs as well. In contrast, YOLOv5 does seem to at least be partially influenced by both the occluded and difficult categories for its occluded and difficult category (as seen in Table 8). Its highest AP classes for the occluded and difficult category are the dining table, chair, sofa, and person classes. Besides the dining table class, the other three were the highest in either occluded or difficult, but not both. In addition, the worst class APs for YOLOv5 in the occluded and difficult category were in the cat, dog, horse, and tv monitor classes. These classes follow in the exact same pattern as the highest classes for this category.

**Table 6 GG model's object class APs for the occluded and difficult, occluded and non-difficult, unoccluded and difficult, and unoccluded and non-difficult breakout categories**

| **GG detected objects (occluded/unoccluded/difficult) Part 2** | | | | |
|---|---|---|---|---|
| **Object class** | **Occluded and difficult** | **Occluded and non-difficult** | **Unoccluded and difficult** | **Unoccluded and non-difficult** |
| aeroplane | Less than 0.01% | 23.82% | 0.17% | 79.45% |
| bicycle | 0.21% | 55.20% | 0.05% | 71.45% |
| bird | Less than 0.01% | 28.66% | 0.03% | 59.53% |
| boat | 0.05% | 3.81% | 1.01% | 43.03% |
| bottle | Less than 0.01% | 6.19% | 0.57% | 31.61% |
| bus | 0.02% | 38.49% | 0.52% | 77.70% |
| car | 0.08% | 30.24% | 0.29% | 69.63% |
| cat | 0.00% | 43.40% | 1.41% | 81.55% |
| chair | 0.37% | 9.84% | 0.39% | 26.56% |
| cow | 0.62% | 26.41% | 0.88% | 44.20% |
| dining table | 6.58% | 31.10% | 18.78% | 25.80% |
| dog | 0.00% | 39.96% | 0.09% | 74.87% |
| horse | 0.10% | 51.07% | 0.41% | 66.63% |
| motorbike | 0.15% | 57.27% | 1.46% | 71.77% |
| person | 0.98% | 51.76% | 1.39% | 63.06% |
| potted plant | 0.94% | 7.29% | 0.70% | 26.49% |
| sheep | 0.03% | 36.56% | 1.46% | 64.00% |
| sofa | 1.73% | 29.56% | 1.91% | 26.93% |
| train | 0.64% | 47.83% | 0.21% | 75.82% |
| tv monitor | Less than 0.01% | 18.07% | 0.04% | 64.76% |

**Table 7 Detectron2’s object class APs for the occluded and difficult, occluded and non-difficult, unoccluded and difficult, and unoccluded and non-difficult breakout categories**

| Detectron2 Faster R-CNN detected objects (occluded/unoccluded/difficult) Part 2 | | | | |
|---|---|---|---|---|
| **Object class** | **Occluded and difficult** | **Occluded and non-difficult** | **Unoccluded and difficult** | **Unoccluded and non-difficult** |
| aeroplane | 0.11% | 36.92% | 12.96% | 88.20% |
| bicycle | 2.50% | 48.15% | 6.25% | 58.64% |
| bird | 0.00% | 37.84% | 0.00% | 66.11% |
| boat | 0.48% | 12.57% | 8.63% | 69.81% |
| bottle | 0.36% | 51.20% | 2.08% | 66.41% |
| bus | 0.00% | 60.91% | 4.17% | 82.55% |
| car | 4.34% | 73.41% | 6.82% | 88.00% |
| cat | 0.00% | 53.04% | 1.11% | 74.90% |
| chair | 3.13% | 35.64% | 0.93% | 62.61% |
| cow | 3.40% | 42.19% | 7.72% | 72.38% |
| dining table | 34.16% | 49.93% | 38.56% | 40.93% |
| dog | 0.00% | 40.07% | 0.00% | 64.53% |
| horse | 0.10% | 78.29% | 1.10% | 82.28% |
| motorbike | 1.40% | 76.68% | 8.06% | 87.48% |
| person | 9.75% | 81.42% | 17.18% | 88.90% |
| potted plant | 0.32% | 27.05% | 1.84% | 56.96% |
| sheep | 2.40% | 62.21% | 3.82% | 77.58% |
| sofa | 6.01% | 54.66% | 5.23% | 56.94% |
| train | 3.33% | 64.19% | 0.00% | 84.49% |
| tv monitor | 0.00% | 30.45% | 6.25% | 66.94% |

**Table 8 YOLOv5's object class APs for the occluded and difficult, occluded and non-difficult, unoccluded and difficult, and unoccluded and non-difficult breakout categories**

| YOLOv5 detected objects (occluded/unoccluded/difficult) Part 2 | | | | |
|---|---|---|---|---|
| Object class | Occluded and difficult | Occluded and non-difficult | Unoccluded and difficult | Unoccluded and non-difficult |
| aeroplane | 0.16% | 5.50% | 0.90% | 88.80% |
| bicycle | 0.24% | 40.00% | 0.23% | 55.00% |
| bird | 0.14% | 11.20% | 0.00% | 75.00% |
| boat | 0.46% | 6.78% | 2.36% | 71.70% |
| bottle | 0.14% | 17.40% | 0.55% | 62.10% |
| bus | 0.25% | 16.60% | 0.23% | 83.80% |
| car | 0.86% | 26.40% | 1.30% | 72.00% |
| cat | 0.00% | 13.50% | 0.49% | 79.70% |
| chair | 1.72% | 24.40% | 1.02% | 49.50% |
| cow | 0.46% | 22.00% | 0.62% | 61.20% |
| dining table | 6.00% | 37.60% | 3.90% | 24.80% |
| dog | 0.00% | 14.10% | 0.10% | 75.80% |
| horse | 0.00% | 39.90% | 0.00% | 55.10% |
| motorbike | 0.23% | 33.80% | 0.44% | 62.20% |
| person | 0.88% | 39.30% | 1.71% | 51.20% |
| potted plant | 0.71% | 14.00% | 0.19% | 48.20% |
| sheep | 0.47% | 24.40% | 1.22% | 64.90% |
| sofa | 1.67% | 39.90% | 2.23% | 38.70% |
| train | 0.39% | 17.20% | 0.00% | 78.90% |
| tv monitor | 0.00% | 11.20% | 0.78% | 73.00% |

## 8. Conclusion

All of the object-detection models used for the experiment work very well in general. When it comes to partially occluded objects, the GG model does well for a few classes but does not work well overall. Detectron2 does the best among the three models, with over half of the classes having at least 40% AP. The occluded AP scores for YOLOv5 are the worst, with only one class that has above 40% AP. All three models falter on objects in poor or abnormal lighting conditions, as indicated by the APs from the difficult category. This is concerning and will need to be improved since variable and uncontrollable lighting conditions are to be expected on the battlefield.

One of the tasks that will be completed soon will be to train and evaluate Detectron2 and YOLOv5 on semantic parts and compare to the GG model, since GG is the only model to natively perform parts detection. Eventually, the models will also have their training augmented by images created from simulation environments to see if their accuracy on real-world objects improves. These images will contain various examples of both partially occluded and difficult objects. Last, some of the attributes from these models will be incorporated into a new, custom parts-detection model that will be developed by researchers within the Battlefield and Information Systems Branch of the Computational and Information Sciences Directorate at the US Army Combat Capabilities Development Command Army Research Laboratory.

## 9. References

1. Gonzalez-Garcia A, Modolo D, Ferrari V. Objects as context for detecting their semantic parts. arXiv.org; 2017 [accessed 2021 Sep 10]. https://arxiv.org/abs/1703.09529.

2. Wu Y, Massa F, Girshick R, Kirillov A, Lo W-Y. Detectron2: A pytorch-based modular object-detection library. Facebook AI Research; 2019 Oct 10 [accessed 2021 Sep 10]. https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/.

3. Jocher G, Stoken A, Chaurasia A, Borovec J, NanoCode012, TaoXie, Kwon Y, Michael K, Changyu L, Fang J, et al. ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support. Ultralytics; 2021 Oct [accessed 2022 Jan 28]. https://github.com /ultralytics/yolov5.

4. Everingham M, van Gool L, Williams C, Winn J, Zisserman A. The PASCAL visual object classes challenge 2010 (VOC 2010) results; 2010 [accessed 2021 Sep 10]. http://host.robots.ox.ac.uk/pascal/VOC/voc2010/.

5. Gonzalez-Garcia A, Modolo D, Ferrari V. Do semantic parts emerge in convolutional neural networks? International Journal of Computer Vision. 2017 Oct;126(5):476–494.

6. Girshick R. Fast R-CNN. arXiv.org; 2015 Sep 27 [accessed 2021 Sep 14]. https://arxiv.org/abs/1504.08083.

7. Ren S, He K, Girshick R, Sun J. Faster R-CNN: towards real-time object-detection with region proposal networks. arXiv.org; 2016 Jan 6 [accessed 2021 Sep 14]. https://arxiv.org/abs/1506.01497.

8. He K, Gkioxari G, Dollar P, Girshick R. Mask R-CNN. arXiv.org; 2018 Jan 24 [accessed 2021 Sep 14]. https://arxiv.org/abs/1703.06870.

9. Lin T-Y, Goyal P, Girshick R, He K, Dollar P. Focal loss for dense object-detection. arXiv.org; 2018 Feb 7 [accessed 2021 Sep 14]. https://arxiv.org /abs/1708.02002.

10. Guler RA, Neverova N, Kokkinos I. DensePose: dense human pose estimation in the wild. arXiv.org; 2018 Feb 1 [accessed 2021 Sep 14]. https:P//arxiv.org /abs/1802.00434.

11. Cai Z, Vasconcelos N. Cascade R-CNN: delving into high quality object-detection. arXiv.org; 2017 Dec 3 [accessed 2021 Sep 14]. https://arxiv.org/abs/1712.00726.

12. Kirillov A, Girshick R, He K, Dollar P. Panoptic feature pyramid networks. arXiv.org; 2019 Apr 10 [accessed 2021 Sep 14]. https://arxiv.org/abs/1901.02446.

13. Chen X, Girshick R, He K, Dollar P. TensorMask: a foundation for dense object segmentation. arXiv.org; 2019 Aug 27 [accessed 2021 Sep 14]. https://arxiv.org/abs/1903.12174.

14. Redmon J. YOLO: Real-time object-detection; 2018 [accessed 2021 Sep 10]. https://pjreddie.com/darknet/yolo/.

15. YOLOv5 Documentation; 2021 [accessed 2022 Jan 28]. https://docs.ultralytics.com.

16. Everingham M, Winn J. devkit_doc.avi; 2010 May 8 [accessed 2021 Sep 10]. http://host.robots.ox.ac.uk/pascal/VOC/voc2010/devkit_doc_08-May-2010.pdf.

## List of Symbols, Abbreviations, and Acronyms

| | |
|---|---|
| AP | average precision |
| CNN | convolutional neural network |
| COCO | Common Objects in Context |
| FC | fully connected |
| GG | Gonzalez–Garcia model |
| GPU | graphics processing unit |
| mAP | mean average precision |
| PASCAL VOC | Pattern Analysis, Statistical Modeling and Computational Learning Visual Object Classes |
| ROI | region of interest |
| XML | extensible markup language |

| | |
|---|---|
| 1<br>(PDF) | DEFENSE TECHNICAL<br>INFORMATION CTR<br>DTIC OCA |
| 1<br>(PDF) | DEVCOM ARL<br>FCDD RLD DCI<br>TECH LIB |
| 1<br>(PDF) | DEVCOM ARL<br>FCDD RLC IB<br>D JEFFERSON II |