IoU a better detection evaluation metric

Choosing the right object detection model means looking at more than just mAP

Eric Hofesmann Aug 24 · 10 min read

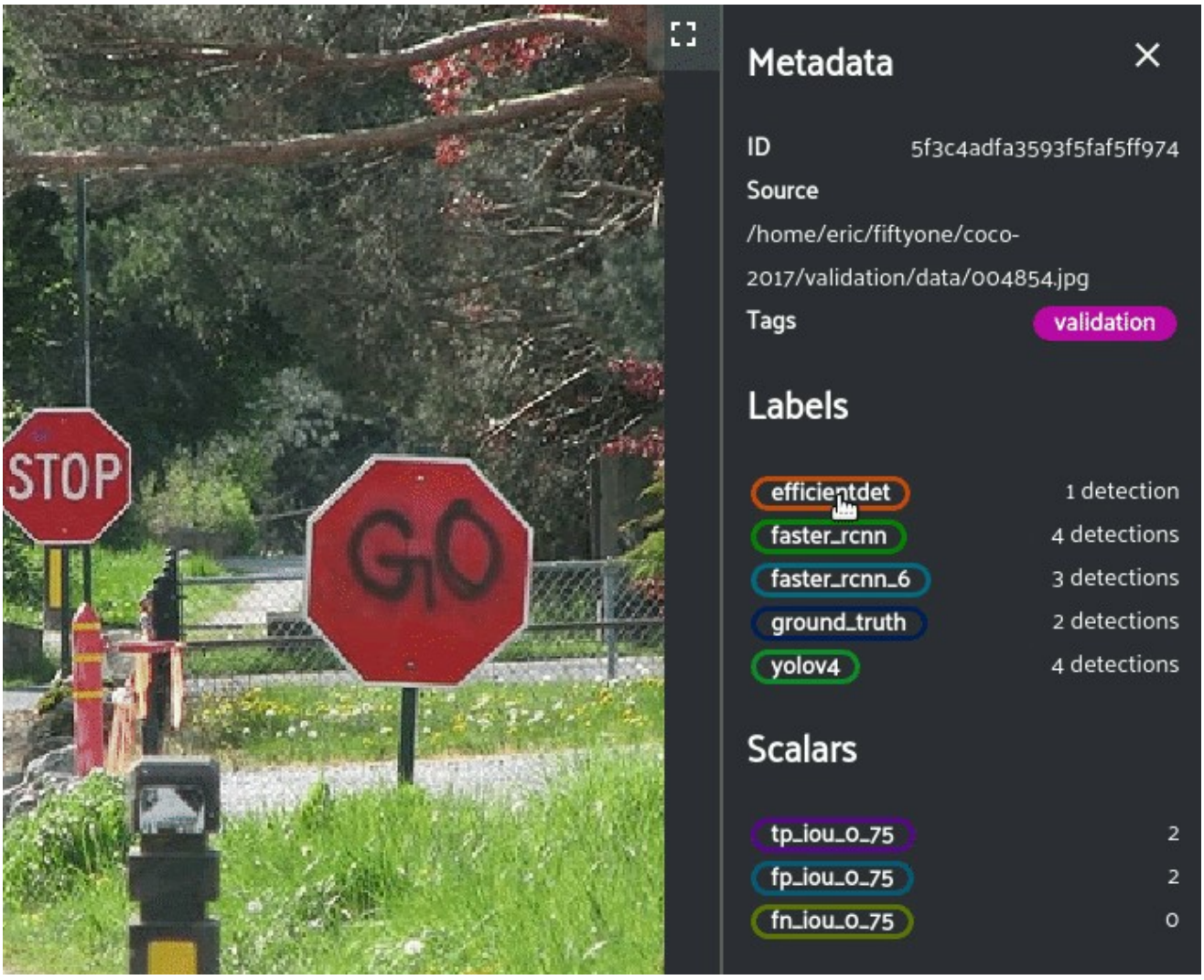
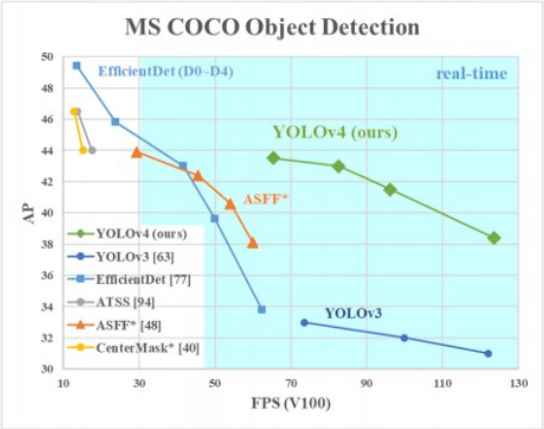


Image from COCO-2017 [1] validation set with detections displayed in FiftyOne [2]

Choosing the best model architecture and pretrained weights for your task can be hard. If you've ever worked on an object detection problem then you've undoubtedly come across plots and tables similar to those below while comparing different models.



Model	test-dev AP	test-dev AP_{50}	test-dev AP_{75}	val AP	Params	Ratio	FLOPs	Ratio	Latency (ms) TitianV	Latency (ms) V100
EfficientDet-D0 (512)	**34.6**	**53.0**	**37.1**	**34.3**	**3.9M**	**1x**	**2.5B**	**1x**	**12**	**10.2**
YOLOv3 [34]	33.0	57.9	34.4	-	-	-	71B	28x	-	-
EfficientDet-D1 (640)	**40.5**	**59.1**	**43.7**	**40.2**	**6.6M**	**1x**	**6.1B**	**1x**	**16**	**13.5**
RetinaNet-R50 (640) [24]	39.2	58.0	42.3	39.2	34M	6.7x	97B	16x	25	-
RetinaNet-R101 (640)[24]	39.9	58.5	43.0	39.8	53M	8.0x	127B	21x	32	-
EfficientDet-D2 (768)	**43.9**	**62.7**	**47.6**	**43.5**	**8.1M**	**1x**	**11B**	**1x**	**23**	**17.7**
Detectron2 Mask R-CNN R101-FPN [1]	-	-	-	42.9	63M	7.7x	164B	15x	-	56‡
Detectron2 Mask R-CNN X101-FPN [1]	-	-	-	44.3	107M	13x	277B	25x	-	103‡
EfficientDet-D3 (896)	**47.2**	**65.9**	**51.2**	**46.8**	**12M**	**1x**	**25B**	**1x**	**37**	**29.0**
ResNet-50 + NAS-FPN (1024) [10]	44.2	-	-	-	60M	5.1x	360B	15x	64	-
ResNet-50 + NAS-FPN (1280) [10]	44.8	-	-	-	60M	5.1x	563B	23x	99	-
ResNet-50 + NAS-FPN (1280@384)[10]	45.4	-	-	-	104M	8.7x	1043B	42x	150	-
EfficientDet-D4 (1024)	**49.7**	**68.4**	**53.9**	**49.3**	**21M**	**1x**	**55B**	**1x**	**65**	**42.8**
AmoebaNet+ NAS-FPN +AA(1280)[45]	-	-	-	48.6	185M	8.8x	1317B	24x	246	-
EfficientDet-D5 (1280)	**51.5**	**70.5**	**56.1**	**51.3**	**34M**	**1x**	**135B**	**1x**	**128**	**72.5**
Detectron2 Mask R-CNN X152 [1]	-	-	-	50.2	-	-	-	-	-	234‡
EfficientDet-D6 (1280)	**52.6**	**71.5**	**57.2**	**52.2**	**52M**	**1x**	**226B**	**1x**	**169**	**92.8**
AmoebaNet+ NAS-FPN +AA(1536)[45]	-	-	-	50.7	209M	4.0x	3045B	13x	489	-
EfficientDet-D7 (1536)	**53.7**	**72.4**	**58.4**	**53.4**	**52M**		**325B**		**232**	**122**
EfficientDet-D7x (1536)	**55.1**	**74.3**	**59.9**	**54.4**	**77M**		**410B**		**285**	**153**

Right image source: YOLOv4 [3]. Left image source: EfficientDet [4]

The main thing that you get out of comparisons like these is which model has a higher mAP on the COCO dataset than other models. But how much does that really mean to you? You need to stop strictly looking at aggregate metrics, look instead at the data and model results in more detail to understand what's working and what's not.

In recent years, great strides are being made to provide similar detection results with faster models, meaning mAP is not the only factor to consider when comparing two detectors. However, no matter how fast your model is, it still needs to provide high-quality detections that meet your requirements.

While it is important to be able to compare different models easily, reducing the performance of a model down to **a single number (mAP) can obscure the intricacies in the model results** that may be important to your problem. You should also be considering:

- Bounding box tightness (IoU)
- High confidence false positives
- Individual samples to spot check performance
- Performance on classes most relevant to your task

What is mAP?

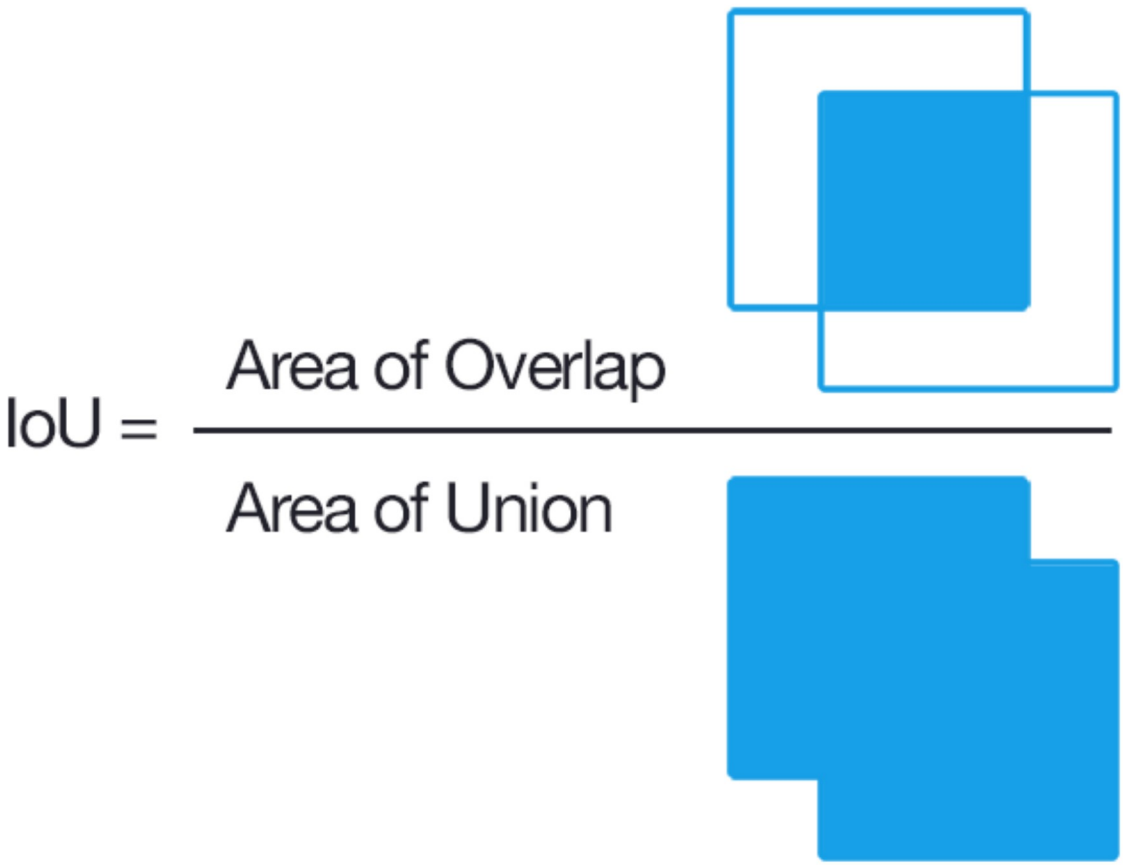
Mean average precision (mAP) is used to determine the accuracy of a set of object detections from a model when compared to ground-truth object annotations of a dataset.

We won't go into full detail here, but you should understand the basics. There is a wide selection of posts discussing mAP in more detail if you are interested [6, 7].

IoU

Intersection over Union (IoU) is used when calculating mAP. It is a number from 0 to 1 that specifies the amount of overlap between the predicted and ground truth bounding box.

- an IoU of 0 means that there is no overlap between the boxes
- an IoU of 1 means that the union of the boxes is the same as their overlap indicating that they are completely overlapping



(source)

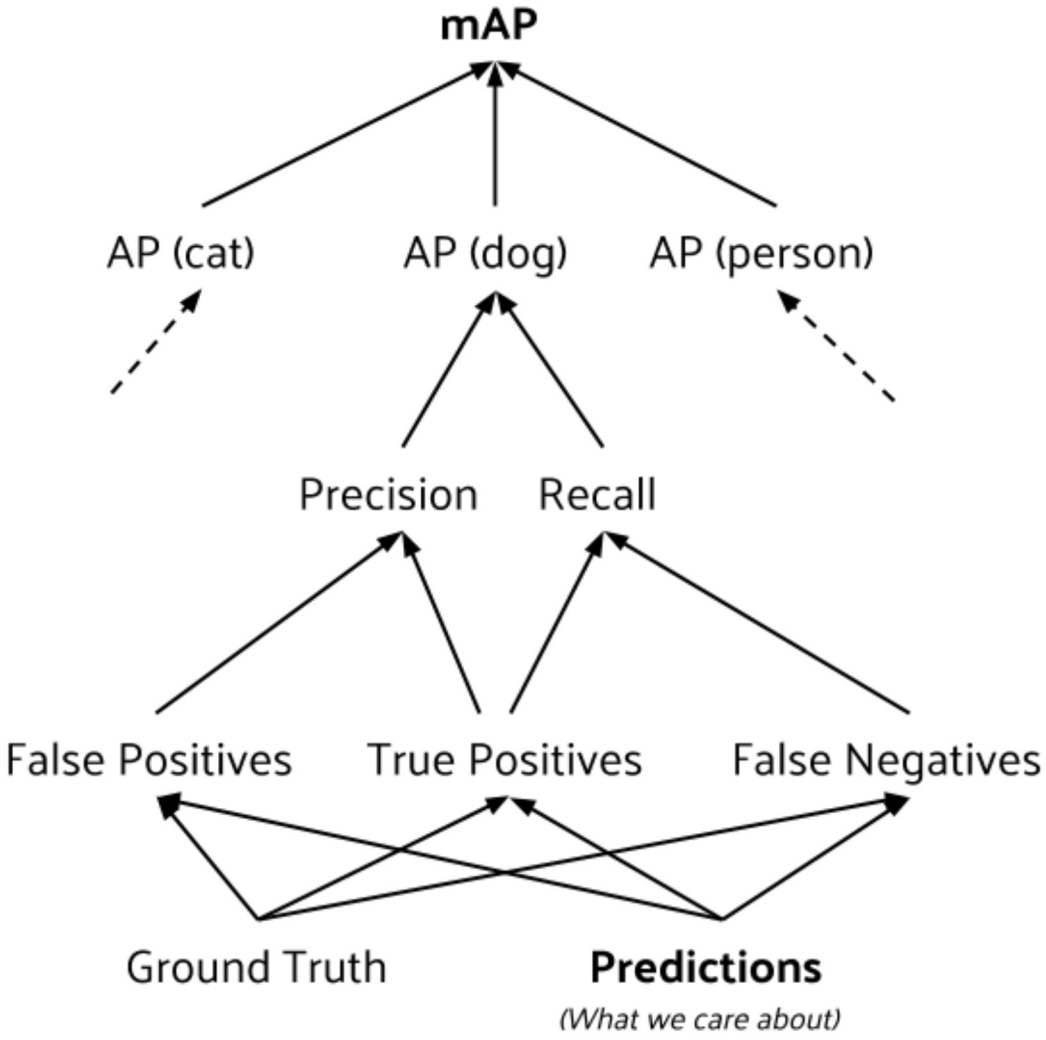
IoU is an important accuracy measure to track when gathering human annotations. The industry best practice is to include a minimum IoU requirement for their human annotation tasks, to ensure that the annotations that are delivered have an IoU >= X (where X = 0.95 is typical) with respect to the "perfect" annotation of that object, as determined by the annotation schema for the project (i.e. box vehicles as tight as possible, including all visible parts of them, including the wheels). State of the are detectors, on the other hand, typically do not perform at a 0.95 IoU as we show in our experiments later in this post.

mAP in short

Mean average precision (mAP) is calculated by first gathering a set of predicted object detections and a set of ground truth object annotations.

- For each prediction, IoU is computed with respect to each ground truth box in the image.
- These IoUs are then thresholded to some value (generally between 0.5 and 0.95) and predictions are matched with ground truth boxes using a greedy strategy (i.e. highest IoUs are matched first).
- A precision-recall (PR) curve is then generated for each object class and the average precision (AP) is computed. A PR curve takes into account the performance of a model with respect to true positives, false positives, and false negatives over a range of confidence values. More details can be found in this post [6].
- The mean of the AP of all object classes is the mAP.
- When evaluating the COCO dataset, this is repeated with 10 IoU thresholds from 0.5 to 0.95 and averaged.

The diagram below demonstrates just how many steps it takes to calculate mAP and how abstract mAP is as a concept when trying to understand the underlying model predictions.



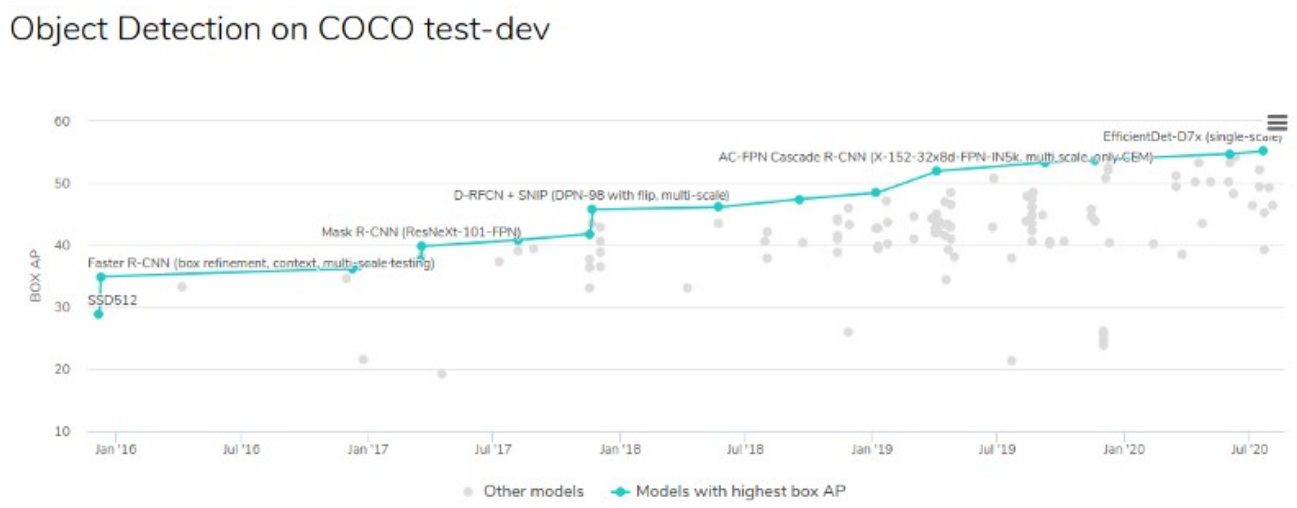
Flow chart showing the steps to relate mAP to model predictions

Why is mAP so popular?

The best answer to why mAP has become the standard for comparing object detection models is because it's convenient. You theoretically only need to use a single number to compare the performance of different models.

If you absolutely need to only use a single criterion for comparison, mAP is a good choice

However, now that the difference in **performance between state-of-the-art models is less than 1% mAP**, there is a dire need for other ways to compare model performance.



Source: Papers with code

Evaluating the ability of a model to localize and classify objects on a per-image basis, particularly its failure modes, provides much deeper insight into the strengths and weaknesses of the model than considering mAP in a vacuum. We call this **atomic evaluation.**

MS COCO cage match

To demonstrate the process of atomic detection evaluation, I compared 3 different object detection models (Faster-RCNN [5], YOLOv4 [3], EfficientDet-D5 [4]) on MSCOCO [1] to see how this evaluation rates them compared to their mAP.

mAP

As a reference, here's the mAP of the models on the COCO-2017 validation set:

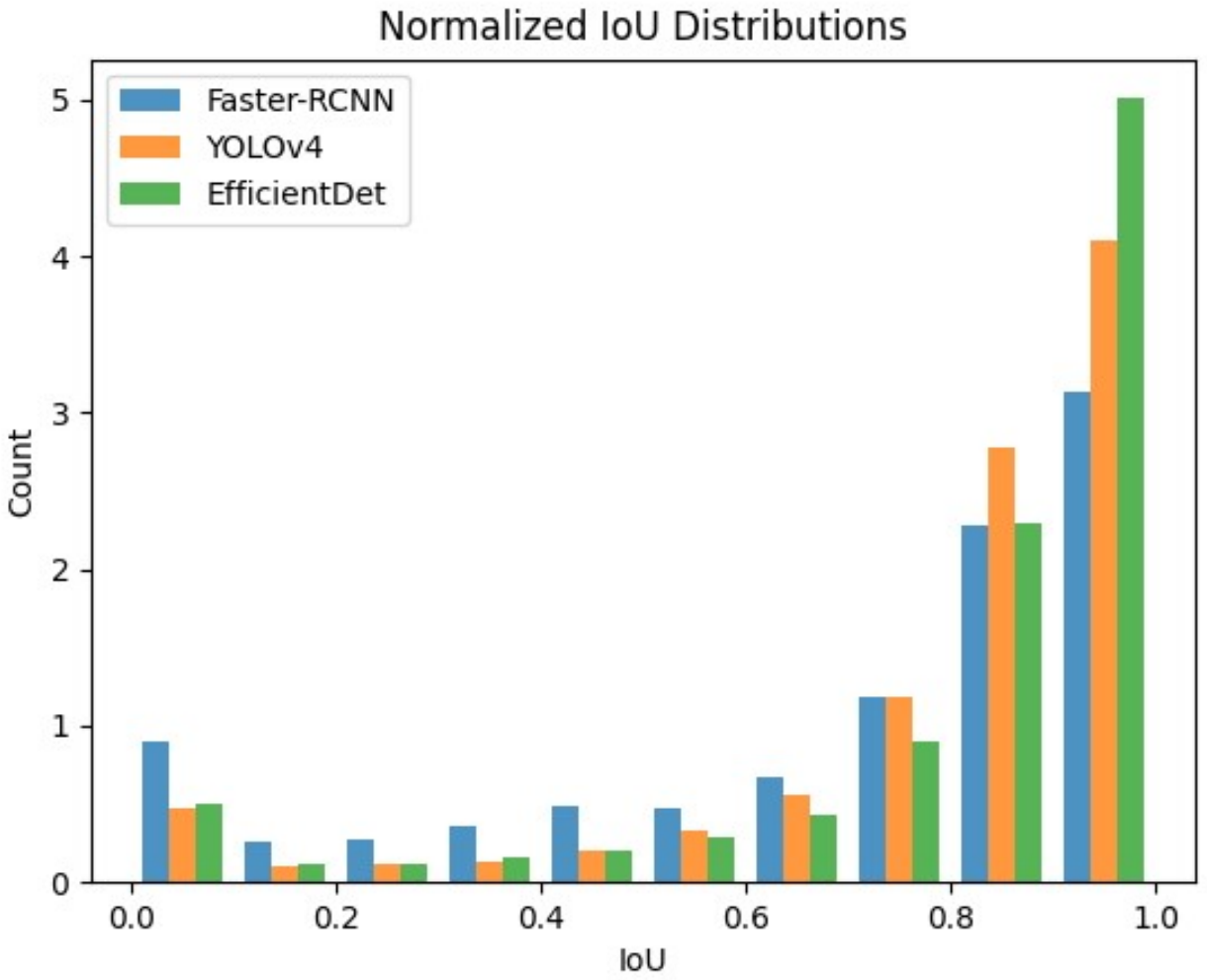
- Faster-RCNN+ResNet50: 33.4% mAP
- EfficientDet-D5 : 41.3% mAP
- YOLOv4: 43.2% mAP

IoU

The mAP of a model alone does not directly show how tight the bounding boxes of your model are because that information is conflated with the correctness of predictions. You must evaluate IoUs directly to know how tight a model's bounding boxes are to the underlying ground truth.

A simple way to generate aggregate statistics about the IoU of different models is by plotting histograms. Here's the basic recipe:

1. Detect objects in a dataset using a set of models



IoU distribution of detection models computed in FiftyOne and visualized with Matplotlib

From these results, we can see that even though the mAP of EfficientDet is lower than YOLOv4 in our experiments, the percentage of bounding boxes with an IoU > 0.9 was higher. This indicates that if you have a task where the tightness of a bounding box is important, EfficientDet is a better choice than either YOLOv4 or Faster-RCNN.

False Positives

A good way to decide which model you should use is to look at the worst-case scenarios and see how each model performs. In this case, we'll look at scenarios where the model

was confident in its prediction but was far off from the ground truth. Specifically, we want to find samples where the model had both:

- High confidence score
- Low IoU

The predictions that meet these criteria generally fall into two categories:

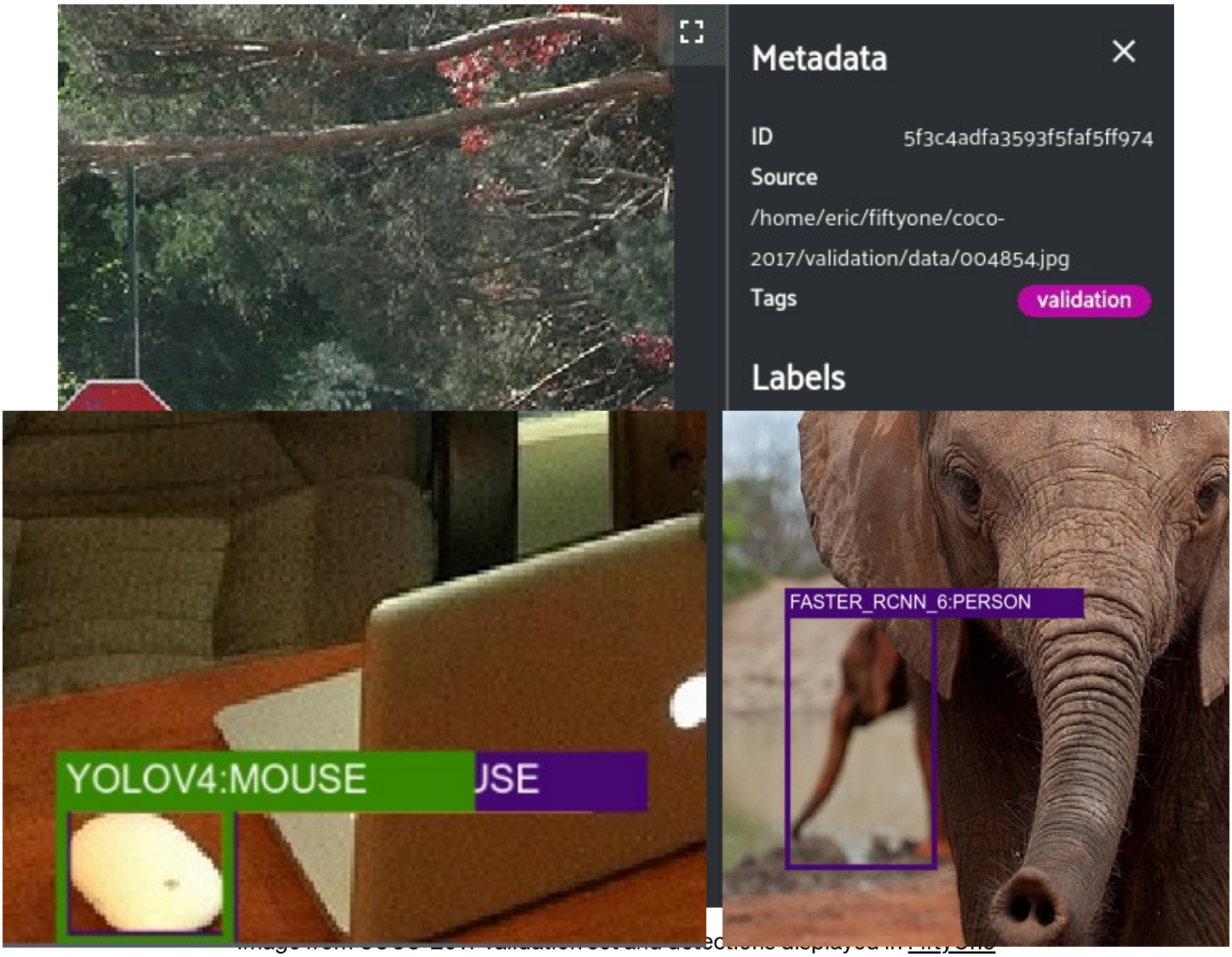
1. They arise from inaccuracies that are built into the model that you are trying gauge
2. They arise from inaccuracies in the ground truth annotations that the model predicted correctly but was marked as false

There is evidence that annotations in popular object detection datasets may not be as correct as you might think. This was demonstrated in this blog post [9] looking at Google's Open Images dataset [10].

I used FiftyOne again to easily find these samples and detections. Once I loaded my model outputs into FiftyOne, I was able to search for examples of where a model, for example, EfficientDet had a maximum IoU < 0.1 with any ground truth object but also had a confidence > 0.9 using the following code:

```python
import fiftyone as fo
import fiftyone.utils.eval as foue
from fiftyone import ViewField as F

dataset = fo.load_dataset("coco-2017-validation")
session = fo.launch_app(dataset=dataset)

foue.evaluate_detections("efficientdet")

# Create a filter after having evaluated my models in FiftyOne
conf_iou_filter = (
    (F("ground_truth_eval.max_iou") < 0.1) &
    (F("confidence") > 0.9)
)

# Filter the dataset for only high confidence low IoU detections
conf_iou_view = dataset.filter_detections(
    "efficientdet", conf_iou_filter
)

# Keep only samples that have a detection matching the above criteria
session.view = conf_iou_view.match(
    F("efficientdet.detections").length() > 0
)
```

fiftyone_eval.py hosted with ❤ by **GitHub** **view raw**



Images from COCO-2017 validation. **(Left)** A mouse predicted with high confidence by all three models but was not annotated. **(Right)** An elephant predicted as "person" with high confidence by Faster-RCNN.

This example popped right out based on the criteria we specified. If you are using a model for an autonomous vehicle task, then you will want to ensure that your model does not suffer from egregious errors like this.

These results indicate that YOLOv4 or EfficientDet are more useful for detecting unlabeled object instances than Faster-RCNN.

Each model had a varying number of samples with predictions falling under our criteria:

- Faster-RCNN: 104 samples
- YOLOv4: 84 samples
- EfficientDet: 5 samples

Spot Checking

After your results look as you might want, it is a good idea to spend some time looking at their detections to build intuition about how they will perform on your task. Examples like those shown in our experiments are not something that could have been uncovered without looking at model outputs. This is where FiftyOne really comes in handy. It allows you to peruse your dataset and detections as closely as you want.

Faster-RCNN contained far more high confidence, low IoU predictions than YOLOv4 and EfficientDet. Visualizing them showed that the few predictions from YOLOv4 and I just the first few dozen samples that I browsed through. I found the example below.

EfficientDet were also predicted by Faster-RCNN. These false positives were generally annotat

Image from COCO-2017 and displayed in FiftyOne. | **Yellow**: Faster-RCNN | **Green**: EfficientDet | **Blue**: YOLOv4

Surprisingly, Faster-RCNN was able to capture more of the people in this image than YOLO and far more than EfficientDet, even though Faster-RCNN has a much lower mAP. This was corroborated by other samples with crowds of people showing the same trend. If you need to be able to detect crowds, then Faster-RCNN may be a better model choice than newer, higher mAP models.

The Winner(s)?

There is no model that is perfect for any task, the best model for you depends on what criteria you have decided and what your end use case is. Between the three models that we have looked at, each shines in different situations in ways that are not elucidated by their mAP. The real winners here are datasets like MSCOCO and tools like FiftyOne, Matplotlib, and many others that allow us to dig in and analyze models quickly and easily before choosing one.

The key message is that it is critical to use the appropriate metric to guide model evaluation and choice based on the specific operation conditions, rather than simply relying on mAP. After analyzing different aspects of model performance, there was no single model that performed best in the above three sections. Each model could be the

right choice depending on the end task, three of which are presented below.

IoU

If your primary concern is the tightness of bounding boxes, then you should probably go with **EfficientDet**. When we plotted all IoUs of each model, EfficientDet had the highest IoU detections.

This could be extremely useful if you are looking to use a model to perform automatic pre-annotation to make your human annotation work more efficient, for example. Drawing and fixing bounding boxes is much more expensive than image or detection-level annotation. You will want your predicted boxes as tight as possible so that annotators can just go through them and decide which to keep and which to throw out without having to spend the time actually changing the boxes.

Quality Assurance

Our experiments show that **YOLOv4** is a well-rounded model. It's bounding box predictions may not be as tight as EfficientDet and it may not work as well in crowds as Faster-RCNN, but it is a close competitor in nearly all situations. Our second experiment showed that predictions from both YOLOv4 and EfficientDet are often correct when their confidence is high.

Take, for example, a case where you are building an object detection dataset and need to iterate over it to ensure your annotations are correct and no objects were missed. From our second experiment, we now know that either EfficientDet or YOLOv4 could be used to provide suggestions of objects that are falsely unlabeled or mislabeled. Since we are just interested in the number of false positives themselves and not the IoU of the detections, YOLOv4 may be a better choice since it is able to run faster than EfficientDet.

Crowds

From seeing multiple examples similar to our last experiment, we saw that **Faster-RCNN** may be the choice for you if you are expecting to detect crowds of objects.

For example, if you are trying to find an object detection model to count the number of people in surveillance footage of New York City, then you don't care about how tight the boxes are or even if they are all correct, you just care about capturing the majority of people in the scene.

References

[1] T. Lin, et al, Microsoft COCO: Common Objects in Context (2014), European Conference in Computer Vision (ECCV)

[2] Voxel51, FiftyOne (2020)

[3] A. Bochkovskiy, et al, YOLOv4: Optimal Speed and Accuracy of Object Detection (2020)

[4] M. Tan, et al, EfficientDet: Scalable and Efficient Object Detection (2020), Conference on Computer Vision and Pattern Recognition (CVPR)

[5] S. Ren, et al, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (2015), Advances in Neural Information Processing Systems (NeurIPS)

[6] J. Hui, mAP (mean Average Precision) for Object Detection (2018), Medium

[7] R. Tan, Breaking Down Mean Average Precision (mAP) (2019), TowardsDataScience

[8] J. Hunter, Matplotlib: A 2D Graphics Environment (2007), Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95

[9] T. Ganter, I performed Error Analysis on Google's Open Images dataset and now I have trust issues (2020), TowardsDataScience

[10] A. Kuznetsova, et al, The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale (2020), International Journal

Thanks to Michael Armanious and Elliot Gunn.

Object Detection Machine Learning Evaluation Fiftyone Visualization