"1^{st} SOLUTION" FOR VIPRIORS OBJECT DETECTION TASK

TECHNICAL REPORT

Zhang Yuqi
Pingan International Smart City
Shenzhen, China
zhangyuqi731@gmail.com

ABSTRACT

The COCO evaluation API is widely adopted to measure the performance of the object detection task. In this report, a misleading bug is reported inside the COCO evaluation code. Using this bug, the performance measured by the COCO evaluation API can be significantly improved but unfortunately the actual detection task is far from solved. If the bug can be revised from the COCO evaluation code, the competitions built on the evaluation metric can be fairer and the winning solutions can be more convincing.

Keywords Object detection · Evaluation metric

1 Description of our submission

The submission file for the score of 0.439 is shown in Figure 1. The results of image 0 and image 1 are shown here and the rest of images follow the same pattern: (1) each image has 10 bounding boxes, (2) each bounding box corresponds to a different class from category id from 1 to 10, (3) each bounding box is [NaN, NaN, NaN, NaN] for its coordinates and an arbitrary value for its score as long as it is in the range of 0 to 1.

It should be noted that the 0.439 is not the highest score this kind of pattern can reach. In the 0.439 submission, only category from 1 to 10 is assigned with bounding box, if all the categories, from 1 to 22 is assigned with bounding box, the evaluation score can be as high as 0.97 (evaluated on the validation set locally and not submitted to the CodaLab server).

The reasons for the high score are: (1) the COCO evaluation code coc treats the "[NaN, NaN, NaN, NaN]" as a valid bounding box and when calculation of the intersection over union (IoU) is made with the ground truth bounding box, the IoU always equals to 1 (with intersection and union equal to NaN, and NaN over NaN equals to 1), so the "[NaN, NaN, NaN, NaN]" is a true positive prediction for any intersection over union from 0.5 to 0.95; (2) every image in the Delft Bike dataset Kayhan et al. [2021] has only one bike in it which makes the ground truth annotation is equipped with at most one bounding box for each class in one image (the ratio of missing parts is not high). From our submission on the CodaLab leaderboard, you can also find the AP at IoU=0.5 is the same as the AP at IoU=0.75 as shown in Figure 2, and this is because every bounding box we predict is either IoU=0 for missing parts or IoU=1 for present parts in each image. It is noticed by many people that the annotations for the Delft Bike dataset has certain noise, the bug revealed in present work makes the submission totally avoids the noise.

2 How did we find this?

The bug in COCO evaluation code is found when finetuning the post process module of the object detector by accident. By accidentally introducing a few NaN bounding boxes into the final detection result, the score of 0.321 is achieved as shown in 3. Therefore only few bounding boxes from the 0.321 submission is NaN and the majority of other bounding boxes are normal boxes. We did not realize this bug until we opened the submission file, carefully look through the file and found the NaN value for few coordinates of bounding box. We still decide to submit the 0.439 submission to the leaderboard because we want to raise the concern of this bug to the community and hope the evaluation code to be used in the future will filter out the NaN value to avoid "accidental and unintentional performance improvement".

```
[{"image_id": 0, "bbox": [NaN, NaN, NaN, NaN], "score": 0.05986642464995384, "category_id": 1},
{"image_id": 0, "bbox": [NaN, NaN, NaN, NaN], "score": 0.5381338000297546, "category_id": 2},
{"image_id": 0, "bbox": [NaN, NaN, NaN, NaN], "score": 0.07654563337564468, "category_id": 3},
{"image_id": 0, "bbox": [NaN, NaN, NaN, NaN], "score": 0.6546739935874939, "category_id": 4},
{"image_id": 0, "bbox": [NaN, NaN, NaN, NaN], "score": 0.37410444021224976, "category_id": 5},
{"image_id": 0, "bbox": [NaN, NaN, NaN, NaN], "score": 0.45144739747047424, "category_id": 6},
{"image_id": 0, "bbox": [NaN, NaN, NaN, NaN], "score": 0.8356177806854248, "category_id": 7},
{"image_id": 0, "bbox": [NaN, NaN, NaN, NaN], "score": 0.16026602685451508, "category_id": 8},
{"image_id": 0, "bbox": [NaN, NaN, NaN, NaN], "score": 0.4450470805168152, "category_id": 9},
{"image_id": 0, "bbox": [NaN, NaN, NaN, NaN], "score": 0.05942821875214577, "category_id": 10},
{"image_id": 1, "bbox": [NaN, NaN, NaN, NaN], "score": 0.07640768587589264, "category_id": 1},
{"image_id": 1, "bbox": [NaN, NaN, NaN, NaN], "score": 0.5401962995529175, "category_id": 2},
{"image_id": 1, "bbox": [NaN, NaN, NaN, NaN], "score": 0.44884905219078064, "category_id": 3},
{"image_id": 1, "bbox": [NaN, NaN, NaN, NaN], "score": 0.6924340724945068, "category_id": 4},
{"image_id": 1, "bbox": [NaN, NaN, NaN, NaN], "score": 0.35409799218177795, "category_id": 5},
{"image_id": 1, "bbox": [NaN, NaN, NaN, NaN], "score": 0.3869372010231018, "category_id": 6},
{"image_id": 1, "bbox": [NaN, NaN, NaN, NaN], "score": 0.8536689877510071, "category_id": 7},
{"image_id": 1, "bbox": [NaN, NaN, NaN, NaN], "score": 0.3640989661216736, "category_id": 8},
{"image_id": 1, "bbox": [NaN, NaN, NaN, NaN], "score": 0.464446097612381, "category_id": 9},
{"image_id": 1, "bbox": [NaN, NaN, NaN, NaN], "score": 0.058993030339479446, "category_id": 10},
```


Figure 1: The submission file for score of 0.439 evaluated at the CodaLab server. Only results for image id 0 and image id 1 are shown here and the results for the rest of images follow the same pattern.

stdout.txt

```
creating index...
index created!
Accumulating evaluation results...
DONE (t=3.22s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.439
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.439
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.439
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.335
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.417
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.324
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.455
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.455
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.455
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.375
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.455
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.455
Metrics written to scores.txt.
```

Figure 2: The stdout download from the CodaLab sever showing the AP at IoU=0.5 equals to AP at IoU=0.75.

#	SCORE	FILENAME	SUBMISSION DATE	STATUS	✔
1	0.28604161	submission.zip	08/03/2021 02:55:40	Finished	
2	0.32127634	submission.zip	09/01/2021 10:01:47	Finished	
3	0.43851398	submission.zip	09/22/2021 03:41:28	Finished	✔

Figure 3: The scores of each submission from CodaLab server.

3 Conclusion

In this report, a bug from the COCO evaluation code is revealed and the detailed analysis of the submission of the score of 0.439 is carried out here. We hope to raise the concern to the community and to improve the evaluation code to filter out NaN bounding boxes in future to make competition of object detection more convincing.

References

https://github.com/cocodataset/cocoapi/blob/master/PythonAPI/pycocotools/cocoeval.py.

Osman Kayhan, Bart Vredebrecht, and Jan van Gemert. DelftBikes, data underlying the publication: Hallucination In Object Detection-A Study In Visual Part Verification. 7 2021. doi:10.4121/14866116.v2. URL https://data.4tu.nl/articles/dataset/DelftBikes_data_underlying_the_publication_Hallucination_In_Object_Detection-A_Study_In_Visual_Part_Verification/14866116.