**YOLOv3: An Incremental Improvement**

Joseph Redmon Ali Farhadi

University of Washington

**Abstract**

1

*We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that’s pretty swell. It’s a little bigger than last time but more accurate. It’s still fast though, don’t worry. At* 320 × 320 *YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves* 57.9 *AP*50 *in 51 ms on a Titan X, com­pared to* 57.5 *AP*50 *in 198 ms by RetinaNet, similar perfor­mance but 3.8*× *faster. As always, all the code is online at*

arXiv:1804.02767v1 [cs.CV] 8 Apr 2018

[https://pjreddie.com/yolo/](https://pjreddie.com/yolo/.)*.*

1. **Introduction**

Sometimes you just kinda phone it in for a year, you know? I didn’t do a whole lot of research this year. Spent a lot of time on Twitter. Played around with GANs a little.

I had a little momentum left over from last year [12] [1]; I managed to make some improvements to YOLO. But, hon­estly, nothing like super interesting, just a bunch of small changes that make it better. I also helped out with other people’s research a little.

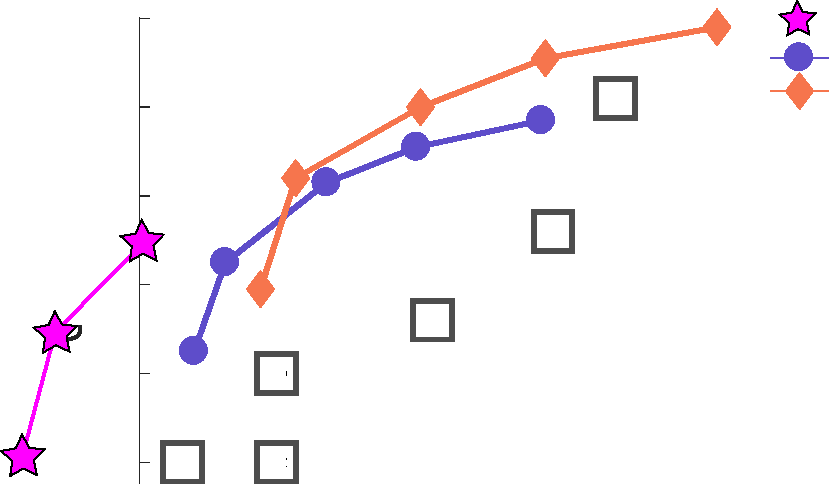
Actually, that’s what brings us here today. We have a camera-ready deadline [4] and we need to cite some of the random updates I made to YOLO but we don’t have a source. So get ready for a TECH REPORT!

The great thing about tech reports is that they don’t need intros, y’all know why we’re here. So the end of this intro­duction will signpost for the rest of the paper. First we’ll tell you what the deal is with YOLOv3. Then we’ll tell you how we do. We’ll also tell you about some things we tried that didn’t work. Finally we’ll contemplate what this all means.

1. **The Deal**

So here’s the deal with YOLOv3: We mostly took good ideas from other people. We also trained a new classifier network that’s better than the other ones. We’ll just take you through the whole system from scratch so you can un­derstand it all.

50 100 150 200 250

inference time (ms)

YOLOv3 RetinaNet-50 RetinaNet-101

**Method mAP time**

1. SSD321 28.0 61
2. DSSD321 28.0 85
3. R-FCN 29.9 85
4. SSD513 31.2 125
5. DSSD513 33.2 156
6. FPN FRCN 36.2 172

RetinaNet-50-500 32.5 73

RetinaNet-101-500 34.4 90

RetinaNet-101-800 **37.8** 198

**YOLOv3-320** 28.2 **22**

**YOLOv3-416** 31.0 29

**YOLOv3-608** 33.0 51

**.000 AP**

38   
36   
34

32

**E**

**F**

**G**

30

**D**

28

**B C**

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

**2.1. Bounding Box Prediction**

Following YOLO9000 our system predicts bounding boxes using dimension clusters as anchor boxes [15]. The network predicts 4 coordinates for each bounding box, tx, ty, tw, th. If the cell is offset from the top left corner of the image by (cx, cy) and the bounding box prior has width and height pw, ph, then the predictions correspond to:

bx = σ(tx) + cx by = σ(ty) + cy bw = pwetw

bh = pheth

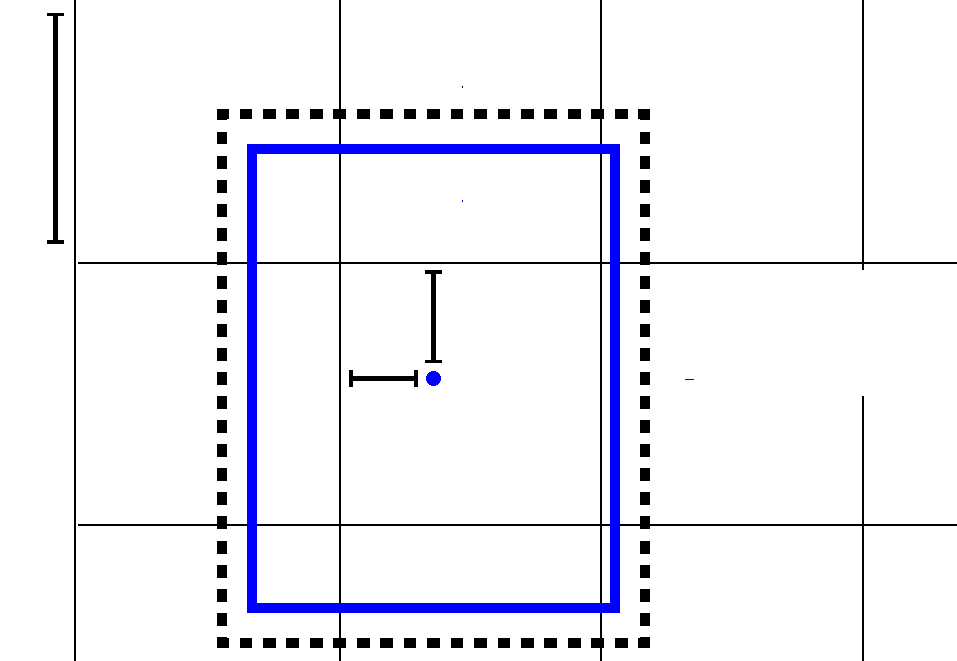
During training we use sum of squared error loss. If the ground truth for some coordinate prediction is ˆt\* our gra­dient is the ground truth value (computed from the ground truth box) minus our prediction: ˆt\* - t\*. This ground truth value can be easily computed by inverting the equations above.

YOLOv3 predicts an objectness score for each bounding box using logistic regression. This should be 1 if the bound­ing box prior overlaps a ground truth object by more than any other bounding box prior. If the bounding box prior

Figure 2. **Bounding boxes with dimension priors and location prediction.** We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function. This figure blatantly self-plagiarized from [15].

cx





cy

ph

bh

σ(ty) σ(tx)

pw

bw

bx=σ(tx)+cx

by

bw=pwe   
bh=phe

=σ(ty)+cy

is not the best but does overlap a ground truth object by more than some threshold we ignore the prediction, follow­ing [17]. We use the threshold of .5. Unlike [17] our system only assigns one bounding box prior for each ground truth object. If a bounding box prior is not assigned to a ground truth object it incurs no loss for coordinate or class predic­tions, only objectness.

**2.2. Class Prediction**

Each box predicts the classes the bounding box may con­tain using multilabel classification. We do not use a softmax as we have found it is unnecessary for good performance, instead we simply use independent logistic classifiers. Dur­ing training we use binary cross-entropy loss for the class predictions.

This formulation helps when we move to more complex domains like the Open Images Dataset [7]. In this dataset there are many overlapping labels (i.e. Woman and Person). Using a softmax imposes the assumption that each box has exactly one class which is often not the case. A multilabel approach better models the data.

**2.3. Predictions Across Scales**

YOLOv3 predicts boxes at 3 different scales. Our sys­tem extracts features from those scales using a similar con­cept to feature pyramid networks [8]. From our base fea­ture extractor we add several convolutional layers. The last of these predicts a 3-d tensor encoding bounding box, ob­jectness, and class predictions. In our experiments with COCO [10] we predict 3 boxes at each scale so the tensor is N x N x [3 \* (4 + 1 + 80)] for the 4 bounding box offsets, 1 objectness prediction, and 80 class predictions.

Next we take the feature map from 2 layers previous and upsample it by 2x. We also take a feature map from earlier in the network and merge it with our upsampled features using concatenation. This method allows us to get more meaningful semantic information from the upsampled fea­tures and finer-grained information from the earlier feature map. We then add a few more convolutional layers to pro­cess this combined feature map, and eventually predict a similar tensor, although now twice the size.

We perform the same design one more time to predict boxes for the final scale. Thus our predictions for the 3rd scale benefit from all the prior computation as well as fine-grained features from early on in the network.

We still use k-means clustering to determine our bound­ing box priors. We just sort of chose 9 clusters and 3 scales arbitrarily and then divide up the clusters evenly across scales. On the COCO dataset the 9 clusters were: (10 x 13), (16 x 30), (33 x 23), (30 x 61), (62 x 45), (59 x 119), (116 x 90), (156 x 198), (373 x 326).

**2.4. Feature Extractor**

We use a new network for performing feature extraction. Our new network is a hybrid approach between the network used in YOLOv2, Darknet-19, and that newfangled residual network stuff. Our network uses successive 3 x 3 and 1 x 1 convolutional layers but now has some shortcut connections as well and is significantly larger. It has 53 convolutional layers so we call it.... wait for it Darknet-53!

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Type | Filters Size | |  | Output | |
|  | Convolutional | 32 | 3 3 |  | 256 | 256 |
|  | Convolutional | 64 | 3 3 | / 2 | 128 | 128 |
|  | Convolutional | 32 | 1 1 |  |  |  |
| 1 | Convolutional | 64 | 3 3 |  |  |  |
|  | Residual |  |  |  | 128 | 128 |
|  | Convolutional | 128 | 3 3 | / 2 | 64 | 64 |
|  | Convolutional | 64 | 1 1 |  |  |  |
| 2 | Convolutional | 128 | 3 3 |  |  |  |
|  | Residual |  |  |  | 64 | 64 |
|  | Convolutional | 256 | 3 3 | / 2 | 32 | 32 |
|  | Convolutional | 128 | 1 1 |  |  |  |
| 8 | Convolutional | 256 | 3 3 |  |  |  |
|  | Residual |  |  |  | 32 | 32 |
|  | Convolutional | 512 | 3 3 | / 2 | 16 | 16 |
|  | Convolutional | 256 | 1 1 |  |  |  |
| 8 | Convolutional | 512 | 3 3 |  |  |  |
|  | Residual |  |  |  | 16 | 16 |
|  | Convolutional | 1024 | 3 3 | / 2 | 8 | 8 |
|  | Convolutional | 512 | 1 1 |  |  |  |
| 4 | Convolutional | 1024 | 3 3 |  |  |  |
|  | Residual |  |  |  | 8 | 8 |
|  | Avgpool |  | Global | |  |  |
| Connected | |  | 1000 |  |  |  |
| Softmax | |  |  |  |  |  |

Table 1. **Darknet-53.**

This new network is much more powerful than Darknet­19 but still more efficient than ResNet-101 or ResNet-152. Here are some ImageNet results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Backbone | Top-1 | Top-5 | Bn Ops | BFLOP/s | FPS |
| Darknet-19 [15] | 74.1 | 91.8 | 7.29 | 1246 | **171** |
| ResNet-101[5] | 77.1 | 93.7 | 19.7 | 1039 | 53 |
| ResNet-152 [5] | **77.6** | **93.8** | 29.4 | 1090 | 37 |
| Darknet-53 | 77.2 | **93.8** | 18.7 | **1457** | 78 |

Table 2. **Comparison of backbones.** Accuracy, billions of oper­ations, billion floating point operations per second, and FPS for various networks.

Each network is trained with identical settings and tested at 256 x 256, single crop accuracy. Run times are measured on a Titan X at 256 x 256. Thus Darknet-53 performs on par with state-of-the-art classifiers but with fewer floating point operations and more speed. Darknet-53 is better than ResNet-101 and 1.5x faster. Darknet-53 has similar perfor­mance to ResNet-152 and is 2x faster.

Darknet-53 also achieves the highest measured floating point operations per second. This means the network struc­ture better utilizes the GPU, making it more efficient to eval­uate and thus faster. That’s mostly because ResNets have just way too many layers and aren’t very efficient.

**2.5. Training**

We still train on full images with no hard negative mining or any of that stuff. We use multi-scale training, lots of data augmentation, batch normalization, all the standard stuff. We use the Darknet neural network framework for training and testing [14].

**3. How We Do**

YOLOv3 is pretty good! See table 3. In terms of COCOs weird average mean AP metric it is on par with the SSD variants but is 3x faster. It is still quite a bit behind other

models like RetinaNet in this metric though.

However, when we look at the “old” detection metric of mAP at IOU= .5 (or AP50 in the chart) YOLOv3 is very strong. It is almost on par with RetinaNet and far above the SSD variants. This indicates that YOLOv3 is a very strong detector that excels at producing decent boxes for ob­jects. However, performance drops significantly as the IOU threshold increases indicating YOLOv3 struggles to get the boxes perfectly aligned with the object.

In the past YOLO struggled with small objects. How­ever, now we see a reversal in that trend. With the new multi-scale predictions we see YOLOv3 has relatively high APS performance. However, it has comparatively worse performance on medium and larger size objects. More in­vestigation is needed to get to the bottom of this.

When we plot accuracy vs speed on the AP50 metric (see figure 5) we see YOLOv3 has significant benefits over other detection systems. Namely, it’s faster and better.

**4. Things We Tried That Didn’t Work**

We tried lots of stuff while we were working on YOLOv3. A lot of it didn’t work. Here’s the stuff we can remember.

**Anchor box** x, y **offset predictions.** We tried using the normal anchor box prediction mechanism where you pre­dict the x, y offset as a multiple of the box width or height using a linear activation. We found this formulation de­creased model stability and didn’t work very well.

Linear x, y predictions instead of logistic. We tried using a linear activation to directly predict the x, y offset instead of the logistic activation. This led to a couple point drop in mAP.

**Focal loss.** We tried using focal loss. It dropped our mAP about 2 points. YOLOv3 may already be robust to the problem focal loss is trying to solve because it has sep­arate objectness predictions and conditional class predic­tions. Thus for most examples there is no loss from the class predictions? Or something? We aren’t totally sure.

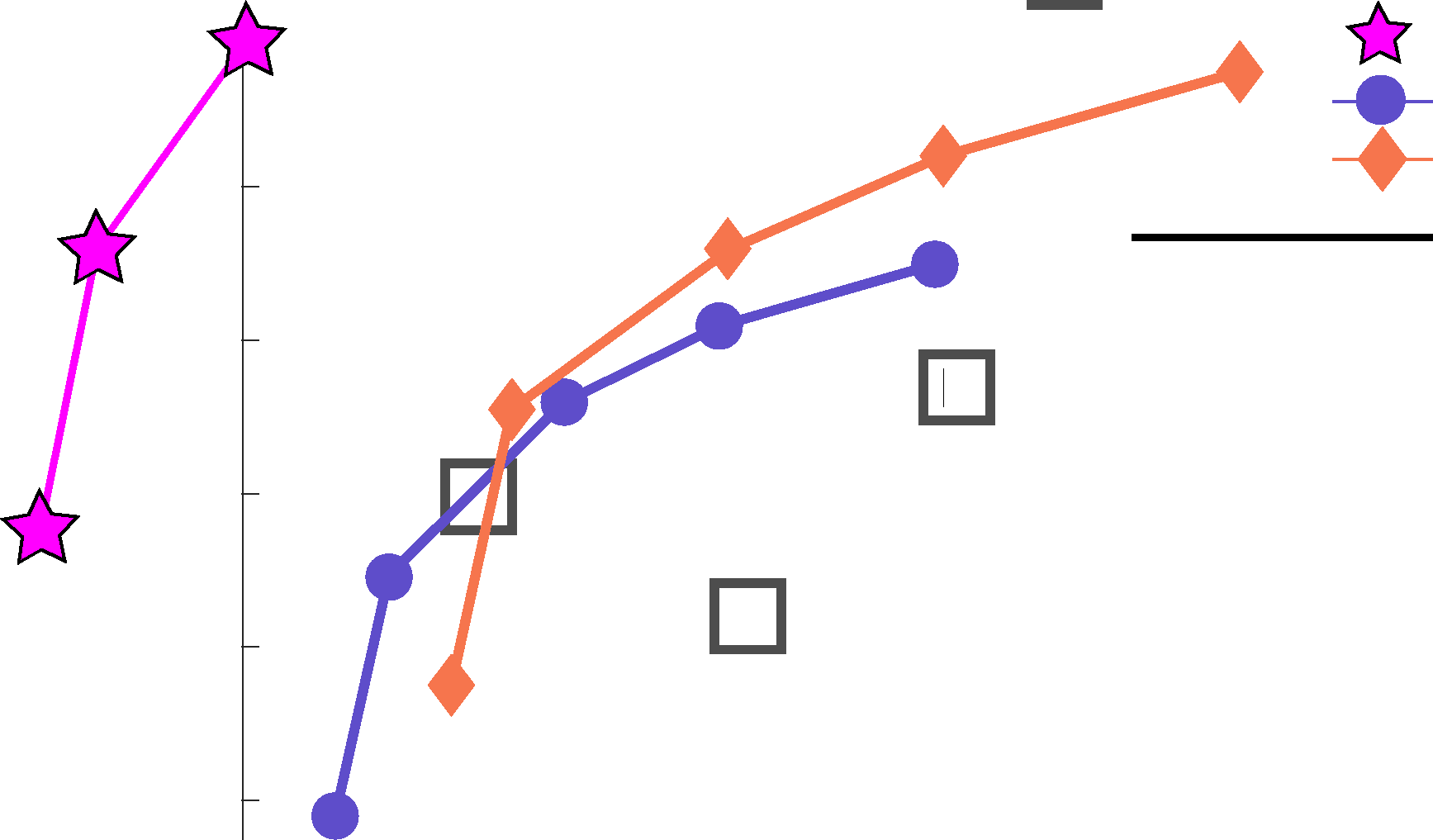
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | backbone | AP | AP50 | AP75 | APS | APM | APL |
| *Two-stage methods* |  |  |  |  |  |  |  |
| Faster R-CNN+++ [5] | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Faster R-CNN w FPN [8] | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Faster R-CNN by G-RMI [6] | Inception-ResNet-v2 [21] | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN w TDM [20] | Inception-ResNet-v2-TDM | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | **52.1** |
| *One-stage methods* |  |  |  |  |  |  |  |
| YOLOv2 [15] | DarkNet-19 [15] | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 |
| SSD513 [11, 3] | ResNet-101-SSD | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 |
| DSSD513 [3] | ResNet-101-DSSD | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 |
| RetinaNet [9] | ResNet-101-FPN | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| RetinaNet [9] | ResNeXt-101-FPN | **40.8** | **61.1** | **44.1** | **24.1** | **44.2** | 51.2 |
| YOLOv3 608 × 608 | Darknet-53 | 33.0 | 57.9 | 34.4 | 18.3 | 35.4 | 41.9 |

Table 3. I’m seriously just stealing all these tables from [9] they take soooo long to make from scratch. Ok, YOLOv3 is doing alright. Keep in mind that RetinaNet has like 3.8× longer to process an image. YOLOv3 is much better than SSD variants and comparable to state-of-the-art models on the AP50 metric.

**C**

|  |  |  |
| --- | --- | --- |
| **Method** | **mAP-50 time** | |
| 1. SSD321 | 45.4 | 61 |
| 1. DSSD321 | 46.1 | 85 |
| 1. R-FCN | 51.9 | 85 |
| 1. SSD513 | 50.4 | 125 |
| 1. DSSD513 | 53.3 | 156 |
| 1. FPN FRCN | **59.1** | 172 |
| RetinaNet-50-500 | 50.9 | 73 |
| RetinaNet-101-500 | 53.1 | 90 |
| RetinaNet-101-800 | 57.5 | 198 |
| **YOLOv3-320** | 51.5 | **22** |
| **YOLOv3-416** | 55.3 | 29 |
| **YOLOv3-608** | 57.9 | 51 |
| 200 | 250 |  |

58   
56



**COCO mAP-JO**

54

**F**

52

**D**

**E**

50

48

YOLOv3 RetinaNet-50 RetinaNet-101

50 100 150

**B C**

inference time (ms)

Figure 3. Again adapted from the [9], this time displaying speed/accuracy tradeoff on the mAP at .5 IOU metric. You can tell YOLOv3 is good because it’s very high and far to the left. Can you cite your own paper? Guess who’s going to try, this guy —+ [16]. Oh, I forgot, we also fix a data loading bug in YOLOv2, that helped by like 2 mAP. Just sneaking this in here to not throw off layout.

**Dual IOU thresholds and truth assignment.** Faster R-CNN uses two IOU thresholds during training. If a predic­tion overlaps the ground truth by .7 it is as a positive exam­ple, by [.3—.7] it is ignored, less than .3 for all ground truth objects it is a negative example. We tried a similar strategy but couldn’t get good results.

We quite like our current formulation, it seems to be at a local optima at least. It is possible that some of these techniques could eventually produce good results, perhaps they just need some tuning to stabilize the training.

**5. What This All Means**

YOLOv3 is a good detector. It’s fast, it’s accurate. It’s not as great on the COCO average AP between .5 and .95 IOU metric. But it’s very good on the old detection metric of .5 IOU.

Why did we switch metrics anyway? The original COCO paper just has this cryptic sentence: “A full discus­sion of evaluation metrics will be added once the evaluation server is complete”. Russakovsky et al report that that hu­mans have a hard time distinguishing an IOU of .3 from .5! “Training humans to visually inspect a bounding box with IOU of 0.3 and distinguish it from one with IOU 0.5 is sur­

prisingly difficult.” [18] If humans have a hard time telling the difference, how much does it matter?

But maybe a better question is: “What are we going to do with these detectors now that we have them?” A lot of the people doing this research are at Google and Facebook. I guess at least we know the technology is in good hands and definitely won’t be used to harvest your personal infor­mation and sell it to.... wait, you’re saying that’s exactly what it will be used for?? Oh.

Well the other people heavily funding vision research are the military and they’ve never done anything horrible like killing lots of people with new technology oh wait 1

I have a lot of hope that most of the people using com­puter vision are just doing happy, good stuff with it, like counting the number of zebras in a national park [13], or tracking their cat as it wanders around their house [19]. But computer vision is already being put to questionable use and as researchers we have a responsibility to at least consider the harm our work might be doing and think of ways to mit­igate it. We owe the world that much.

In closing, do not @ me. (Because I finally quit Twitter).

1The author is funded by the Office of Naval Research and Google.

**References**

1. Analogy. *Wikipedia*, Mar 2018. 1
2. M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) chal­lenge. *International journal of computer vision*, 88(2):303– 338, 2010. 6
3. C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017. 3
4. D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. Iqa: Visual question answering in interactive environments. *arXiv preprint arXiv:1712.03316*, 2017. 1
5. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learn­ing for image recognition. In *Proceedings of the IEEE con­ference on computer vision and pattern recognition*, pages 770–778, 2016. 3
6. J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. 3
7. I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik,

D. Cai, Z. Feng, D. Narayanan, and K. Murphy. Open-images: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from* [*https://github.com/openimages*](https://github.com/openimages,), 2017. 2

1. T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017. 2, 3
2. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Doll´ar. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017. 1, 3, 4
3. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ra­manan, P. Doll´ar, and C. L. Zitnick. Microsoft coco: Com­mon objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2
4. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.- Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 3
5. I. Newton. *Philosophiae naturalis principia mathematica*. William Dawson & Sons Ltd., London, 1687. 1
6. J. Parham, J. Crall, C. Stewart, T. Berger-Wolf, and

D. Rubenstein. Animal population censusing at scale with citizen science and photographic identification. 2017. 4

1. J. Redmon. Darknet: Open source neural networks in c. [http://pjreddie.com/darknet/](http://pjreddie.com/darknet/,), 2013–2016. 3
2. J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6517–6525. IEEE, 2017. 1, 2, 3
3. J. Redmon and A. Farhadi. Yolov3: An incremental improve­ment. *arXiv*, 2018. 4
4. S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: To­wards real-time object detection with region proposal net­works. *arXiv preprint arXiv:1506.01497*, 2015. 2
5. O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2121–2131, 2015. 4
6. M. Scott. Smart camera gimbal bot scanlime:027, Dec 2017. 4
7. A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Be­yond skip connections: Top-down modulation for object de­tection. *arXiv preprint arXiv:1612.06851*, 2016. 3
8. C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. 2017. 3

0 50 100 150 200 0 12.5 25 37.5 50

100 100

75 75



YOLOv3 All the other slow ones

**mAP 50**

50

All the other slow ones

YOLOv3

50

**mAP 50**

25 25

0 0

Execution time (ms) FPS

Figure 4. Zero-axis charts are probably more intellectually honest... and we can still screw with the variables to make ourselves look good!

**Rebuttal**

We would like to thank the Reddit commenters, labmates, emailers, and passing shouts in the hallway for their lovely, heart­felt words. If you, like me, are reviewing for ICCV then we know you probably have 37 other papers you could be reading that you’ll invariably put off until the last week and then have some legend in the field email you about how you really should finish those re­views execept it won’t entirely be clear what they’re saying and maybe they’re from the future? Anyway, this paper won’t have be­come what it will in time be without all the work your past selves will have done also in the past but only a little bit further forward, not like all the way until now forward. And if you tweeted about it I wouldn’t know. Just sayin.

Reviewer #2 AKA Dan Grossman (lol blinding who does that) insists that I point out here that our graphs have not one but two non-zero origins. You’re absolutely right Dan, that’s because it looks way better than admitting to ourselves that we’re all just here battling over 2-3% mAP. But here are the requested graphs. I threw in one with FPS too because we look just like super good when we plot on FPS.

Reviewer #4 AKA JudasAdventus on Reddit writes “Entertain­ing read but the arguments against the MSCOCO metrics seem a bit weak”. Well, I always knew you would be the one to turn on me Judas. You know how when you work on a project and it only comes out alright so you have to figure out some way to justify how what you did actually was pretty cool? I was basically trying to do that and I lashed out at the COCO metrics a little bit. But now that I’ve staked out this hill I may as well die on it.

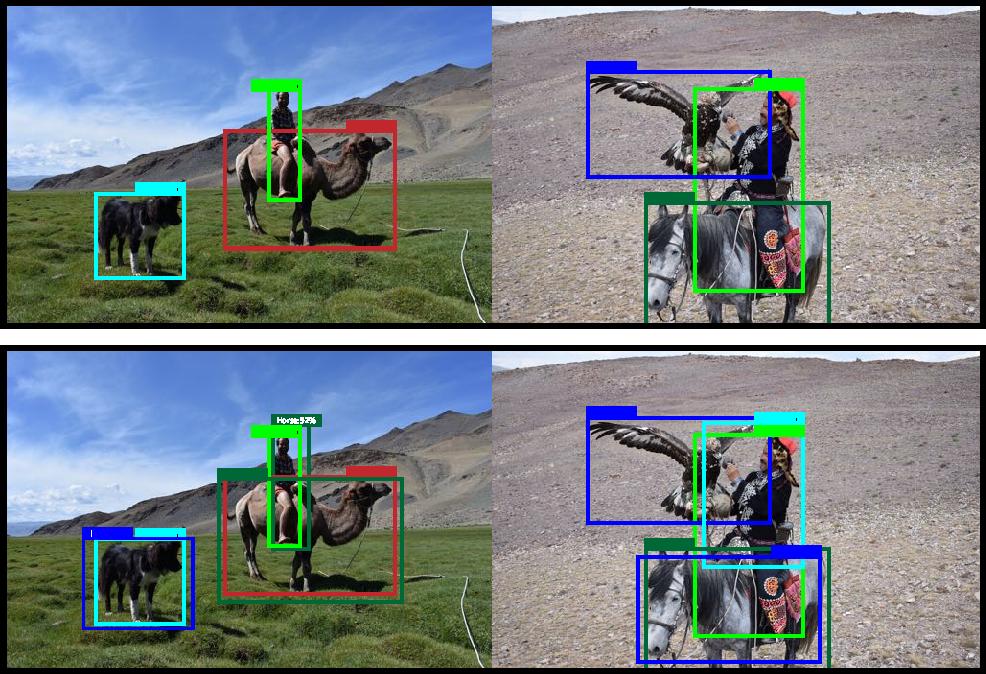
See here’s the thing, mAP is already sort of broken so an up­date to it should maybe address some of the issues with it or at least justify why the updated version is better in some way. And that’s the big thing I took issue with was the lack of justification. For PASCAL VOC, the IOU threshold was ”set deliberately low to ac­count for inaccuracies in bounding boxes in the ground truth data“ [2]. Does COCO have better labelling than VOC? This is defi­nitely possible since COCO has segmentation masks maybe the labels are more trustworthy and thus we aren’t as worried about inaccuracy. But again, my problem was the lack of justification.

The COCO metric emphasizes better bounding boxes but that emphasis must mean it de-emphasizes something else, in this case classification accuracy. Is there a good reason to think that more

precise bounding boxes are more important than better classifi­cation? A miss-classified example is much more obvious than a bounding box that is slightly shifted.

mAP is already screwed up because all that matters is per-class rank ordering. For example, if your test set only has these two images then according to mAP two detectors that produce these results are JUST AS GOOD:

Figure 5. These two hypothetical detectors are perfect according to   
mAP over these two images. They are both perfect. Totally equal.



Detector #1

Detector #2

Bird: 89%

Dog: 99%

Dog: 48%

Horse: 60%

Person: 99%

Person: 42%

Camel: 99%

Camel: 10%

Bird: 99%

Bird: 90%

Horse: 99%

Horse: 70%

Person: 99%

Person: 11%

Dog: 45%

Bird: 75%

Now this is OBVIOUSLY an over-exaggeration of the prob­lems with mAP but I guess my newly retconned point is that there are such obvious discrepancies between what people in the “real world” would care about and our current metrics that I think if we’re going to come up with new metrics we should focus on these discrepancies. Also, like, it’s already mean average preci­sion, what do we even call the COCO metric, average mean aver­age precision?

Here’s a proposal, what people actually care about is given an image and a detector, how well will the detector find and classify objects in the image. What about getting rid of the per-class AP and just doing a global average precision? Or doing an AP calcu­lation per-image and averaging over that?

Boxes are stupid anyway though, I’m probably a true believer in masks except I can’t get YOLO to learn them.