Bag of Freebies for Training Object Detection Neural Networks

Zhi Zhang, Tong He, Hang Zhang, Zhongyuan Zhang, Junyuan Xie, Mu Li
Amazon Web Services
{zhiz, htong, hzaws, zhongyue, junyuanx, mli}@amazon.com

Abstract

Comparing with enormous research achievements targeting better image classification models, efforts applied to object detector training are dwarfed in terms of popularity and universality. Due to significantly more complex network structures and optimization targets, various training strategies and pipelines are specifically designed for certain detection algorithms and no other. In this work, we explore universal tweaks that help boosting the performance of state-of-the-art object detection models to a new level without sacrificing inference speed. Our experiments indicate that these freebies can be as much as 5% absolute precision increase that everyone should consider applying to object detection training to a certain degree.

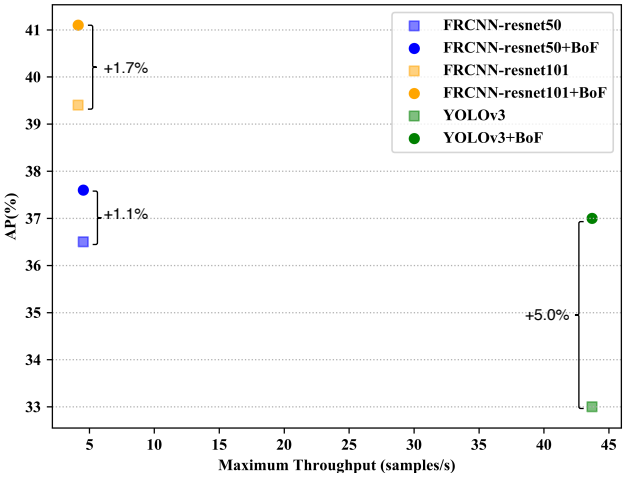

Figure 1. **Bag of Freebies dramatically improves object detector performances with no throughput penalty**.

1. Introduction

Object detection is no doubt one of the cutting edge applications in computer vision drawing attentions of researchers from various fields. Latest state-of-the-art detectors, including single (SSD [12] and YOLO [16]) or multiple stage RCNN-like [17] neural networks, are based on image classification backbone networks, e.g., VGG [20], ResNet [7], Inception [21] and MobileNet series [9, 19].

However, due to the relatively higher model capacity and training complexity, object detection attracted inferior attentions and benefit less from recent researches of training tweaks. What made things worse, is that different detection networks are cherry-picking their own training pipelines without explicit initialization, data pre-processing and optimization analysis, resulting in massive chaos in the adoption of latest techniques proved to lift image classification task.

In this work, we focus on exploring effective approaches that can boost the performance of popular object detection networks without introducing extra computational cost. We first explore the mixup technique on object detection. Unlike [23], we recognize the special property of multiple object detection task which favors spatial preserving transforms, and thus proposed a visually coherent image mixup methods for object detection tasks. Second, we explore detailed training pipelines including learning rate scheduling, weight decay and synchronized BatchNorm. Third, we investigate the effectiveness of our training tweaks by incrementally stacking them to train single and multiple stage object detection networks.

Our major contributions can be summarized as follows:

1) We are the first to systematically evaluate the various training heuristics applied in different object detection pipelines, providing valuable practice guidelines for future researches.

2) We proposed a visually coherent image mixup method designed for training object detection networks which is proved to be very effective in improving model generalization capabilities.

3) We achieved up to 5% out of 30% absolute average precision improvement based on existing models without modifying network structure and the loss function. What we achieved is all free lunch with no extra inference cost.

4) We have extended the research depth on object detection data augmentation domain that significantly strengthened the model generalization capability and help reduce over-fitting problems. The experiments also unveiled good

techniques that can boost object detection performance consistently across different network structures.

The rest of this paper will be organized as follows. First, we will briefly introduce previous works in Section 2 on improving image classification and the potential to transfer to object detection models. Second, the proposed tweaks are detailed in Section 3. Third, the experimental results will be benchmarked in Section 4. Finally, Section 5 will conclude this work.

All related code are open-sourced and pre-trained weights for the models are available in GluonCV toolkit [1].

2. Related Work

In this section, we briefly discuss related work regarding bag of tricks for image classification and heuristic object detection in common.

2.1. Scattering tricks from Image Classification

Image classification serves as the foundation of all most all computer vision tasks. Classification models are cheap compared with popular object detection and semantic segmentation models, therefore attractive enormous researchers to prototyping ideas. In this section, we briefly introduce previous work that opened the shed for this work. Learning rate warm up heuristic [6] was introduced to overcome the negative effect of extremely large mini-batch size. Interestingly, even though mini-batch size used in typical object detection training is nowhere close to the scale in image classification(*e.g*. 10k or 30k [6]), a large amount of anchor size(up to 30k) is effectively contributing to batch size implicitly. a gradual warmup heuristic is crucial to YOLOv3 [16] as in our experiments. There is a line of approaches trying to address the vulnerability of deep neural network. Label smoothing was introduced in [21], which modifies the hard ground truth labeling in cross entropy loss. Zhang *et al*. [23] proposed *mixup* to alleviate adversarial perturbation. Cosine annealing strategy for learning rate decay is proposed in [13] in response to traditional step policy. He *et al*. achieved significant improvements on training accuracy by exploring bag of tricks [8]. In this work, we dive deeper into the heuristic techniques introduced by image classification in the context of object detection.

2.2. Deep Object Detection Pipelines

Most state-of-the-art deep neural network based object detection models are derived from multiple stages and single stage pipelines, starting from R-CNN [4] and YOLO [15], respectively. In single stage pipelines, predictions are generated by a single convolutional network and therefore preserve the spatial alignments (except that YOLO used Fully Connected layers at the end). However, in multiple stage pipelines, e.g. Fast R-CNN [3] and Faster-RCNN [17], final predictions are generated from features which were sampled and pooled in a specific region of interests (RoIs). RoIs were either propagated by neural networks or deterministic algorithms (e.g. Selective Search [22]). This major difference caused significant divergence in data processing and network optimization. For example, due to the lack of spatial variation in single stage pipelines, spatial data augmentation is crucial to the performance as proven in Single-Shot MultiBox Object Detector (SSD) [12]. Due to lack of exploration, many training details are exclusive to one series. In this work, we systematically explore the mutually beneficial tweaks and tricks that may help to boost the performance for both pipelines.

3. Technique Details

In this section, we propose a visual coherent image mixup method for object detection and introduce data processing and training scheduler designed for systematically improving the model performance of object detectors.

3.1. Visually Coherent Image Mixup for Object Detection

Mixup, introduced by Zhang *et al*. [23] is proved to be successful in alleviating adversarial perturbations in classification networks. The distribution of blending ratio in mix-up algorithm proposed by Zhang *et al*. [23] is drawn from beta distribution with $(a = 0.2, b = 0.2)$. The majority of mix-ups are barely noises with such beta distribution. Inspired by the heuristic experiments in Rosenfeld *et al*. [18], we focus on the natural co-occurrence object presentations which play significant roles in object detection. The semi-adversarial object patch transplantation method is not a traditional attack. By applying more complex spatial transformed, we introduced occlusions, spatial signal perturbations that are common in natural image presentations.

In our empirical experiments, continue increasing blending ratio used in the mixup, the objects in resulting frames are more vibrant and coherent to the natural presentations, similar to the transition frames commonly in low FPS movies. The visual comparisons of image classification and such high ratio mixup are illustrated in Fig. 2 and Fig. 3, respectively. In particular, we use geometry preserved alignment for image mixup to avoid distort images at the initial steps. We also choose a beta distribution with more visually coherent ratios $a >= 1$ and $b >= 1$ instead of following the same practice in image classification, as depicted in Figure 4.

We also experimentally tested empirical mixup ratio distributions using the YOLOv3 network on Pascal VOC dataset. Table. 1 shows the actual improvements by adopting detection mixups. Beta distribution with α and β both equal to 1.5 is marginally better than 1.0 (equivalent to uniform distribution) and better than fixed even mixup.

Figure 2. Mix-up visualization of image classification with typical mixup ratio at $0.1 : 0.9$.

Figure 3. Geometry preserved alignment of mixed images for object detection. Image pixels are blended, object labels are merged with respect to generated new image.

To validate the effectiveness of visually coherent mixup, we followed the same experiments of "Elephant in the room" [18] by sliding an elephant image patch through an indoor room image. We trained two YOLOv3 models on COCO 2017 dataset with identical settings except for that model **mix** is using our mixup approach. We depict some surprising discoveries in Fig. 5. As we can observe in Fig. 5, vanilla model trained without our **mix** approach is struggles to detect "elephant in the room" due to heavy occlusion and lack of context because it's rare to capture an elephant in a kitchen. Actually, there is no such training image after examine the common training datasets. In comparison, models trained with our **mix** approach is more robust thanks to randomly generated visually deceptive training images. In addition, we also notice that **mix** model is more *humble*, less confident and generates lower scores for objects on average. However, this behavior does not affect evaluation results as we will show in the experiment section.

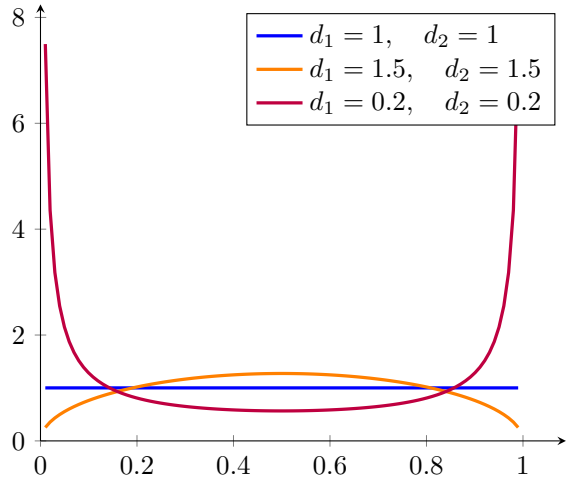

Figure 4. Comparison of different random weighted mix-up sampling distributions. Red curve indicate the typical mixup ratio used in image classification.

3.2. Classification Head Label Smoothing

For each object, detection networks often compute a probability distribution over all classes with softmax function:

$$p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}, \tag{1}$$

where z_is are the unnormalized logits directly from the last linear layer for classification prediction. For object detection during training, we only modify the classification loss by comparing the output distribution p against the ground truth distribution q with cross-entropy

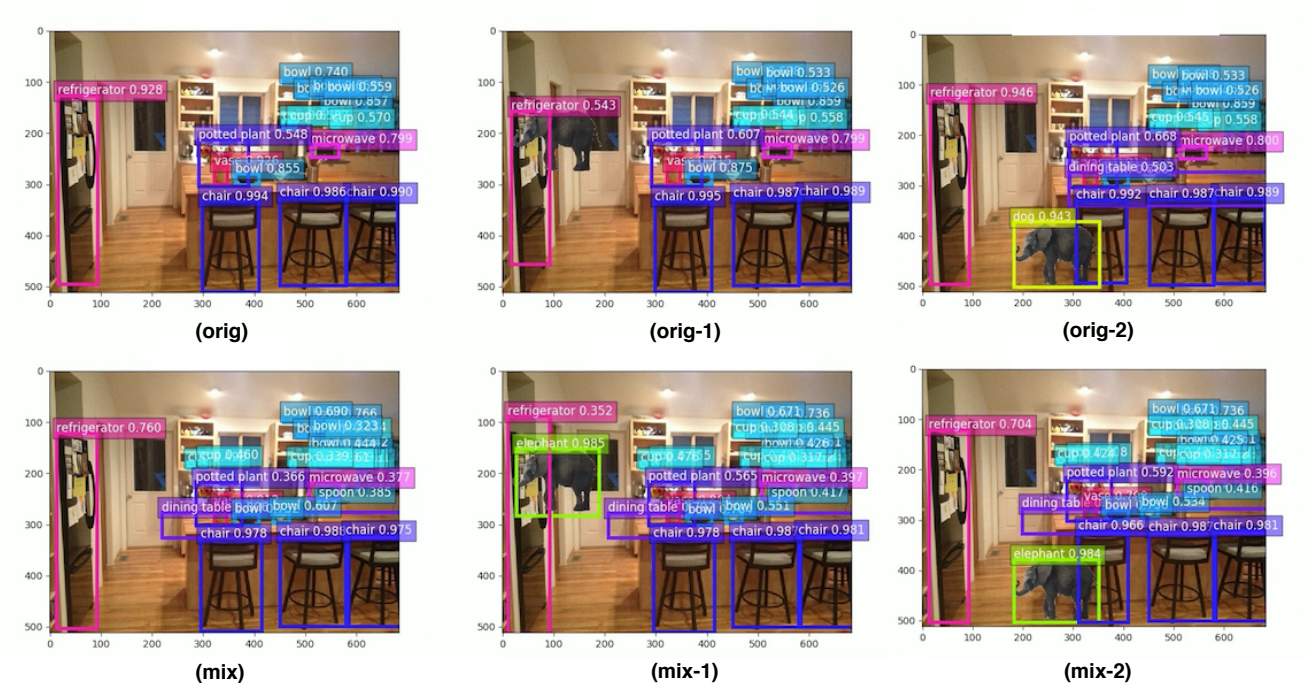

Figure 5. Elephant in the room example. Model trained with geometry preserved mixup (bottom) is more robust against alien objects compared to baseline (top).

Model	mAP @ 0.5
baseline	81.5
0.5:0.5 evenly	83.05
beta(1.0, 1.0), weighted loss	83.48
beta(1.5, 1.5), weighted loss	83.54

Table 1. Effect of various mix-up approaches, validated with YOLOv3 [16] on Pascal VOC 2007 test set. **Weighted loss** indicates the overall loss is the summation of multiple objects with ratio 0 to 1 according to image blending ratio they belong to in the original training images.

$$L = \sum_i q_i \log p_i. \tag{2}$$

q is often a one-hot distribution, where the correct class has probability one while all other classes have zero. Softmax function, however, can only approach this distribution when $z_i \gg z_j, \forall j \neq i$ but never reach it. This encourages the model to be too confident in its predictions and is prone to over-fitting.

Label smoothing was proposed by Szegedy *et al*. [21] as a form of regularization. We smooth the ground truth distribution with

$$q_i' = (1 - \epsilon)q_i + \frac{\epsilon}{K}, \tag{3}$$

where K is the total number of classes and ϵ is a small constant. This technique reduces the model's confidence, measured by the difference between the largest and smallest logits.

In the case of sigmoid outputs in 0 to 1.0 as in YOLOv3 [16], label smoothing is even simpler by correcting the upper and lower limit of the range of targets as in Eq. 3.

3.3. Data Pre-processing

Unlike image classification domain, where the network is extremely tolerant to image geometrical transformation. It is actually encouraged to do so in order to improve generalization accuracy. However, for object detection image pre-processing, we need to carry additional cautious since detection networks are more sensitive to such transformations.

We experimentally reviewed the following data augmentation methods:

- Random geometry transformation. Including random cropping (with constraints), random expansion, random horizontal flip and random resize (with random interpolation).
- Random color jittering including brightness, hue, saturation, and contrast.

In terms of types of detection networks, there are two pipelines for generating final predictions. First is single stage detector network, where final outputs are generated from every single pixel on feature map, for example,

SSD[12] and YOLO[16] networks which generate detection results proportional to spatial shape of an input image. The second is multi-stage proposal and sampling based approached, following Fast-RCNN[17], a large amount of ROI candidates are generated and sampled with a fixed number, the detection results are produced by repeatedly cropping the corresponding region on feature maps and the number of predictions is proportional to the fixed sampling number.

Since sampling-based approaches repeat enormous crop like operations on feature maps, it substitutes the operation of randomly cropping the input image, therefore these networks do not require extensive geometric augmentations applied during the training stage.

3.4. Training Scheduler Revamping

During training, the learning rate usually starts with a relatively big number and gradually becomes smaller throughout the training process. For example, the step schedule is the most widely used learning rate schedule. With step schedule, the learning rate is multiplied by a constant number below 1 after reaching pre-defined epochs or iterations. For instance, the default step learning rate schedule for Faster-RCNN [17] is to reduce learning rate by ratio 0.1 at $60k$ iterations. Similarly, YOLOv3 [16] uses same ratio 0.1 to reduce learning rate at $40k$ and $45k$ iterations. Step scheduler has sharp learning rate transition which may cause the optimizer to re-stabilize the learning momentum in the next few iterations. In contrast, a smoother cosine learning rate adjustment was proposed by Loshchilov *et al*. [13]. Cosine schedule scales the learning rate according to the value of cosine function on 0 to pi. It starts with slowly reducing large learning rate, then reduces the learning rate quickly halfway, and finally ends up with tiny slope reducing small learning rate until it reaches 0.

Warm up learning rate is another common strategy to avoid gradient explosion during the initial training iterations. Warm-up learning rate schedule is critical to several object detection algorithms, e.g., YOLO v3, which has a dominant gradient from negative examples in the very beginning iterations where sigmoid classification score is initialized around 0.5 and biased towards 0 for the majority predictions.

Training with cosine schedule and proper warmup lead to better validation accuracy, as depicted in Fig. 6, validation mAP achieved by applying cosine learning rate decay outperforms step learning rate schedule at all times in training. Due to the higher frequency of learning rate adjustment, it also suffers less from plateau phenomenon of step decay that validation performance will be stuck for a while until learning rate is reduced.

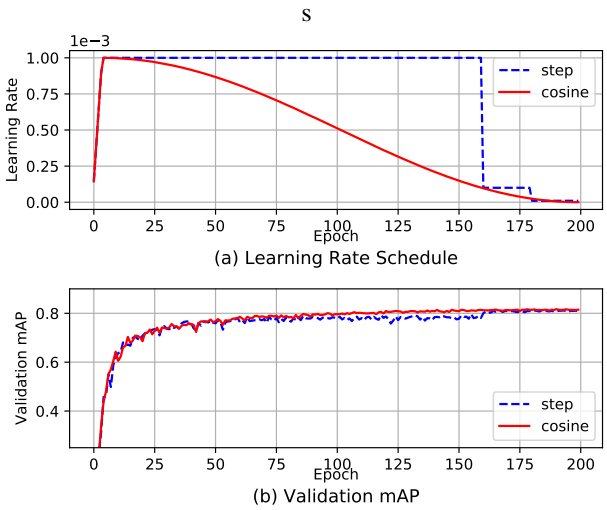

Figure 6. Visualization of learning rate scheduling with warm-up enabled for YOLOv3 training on Pascal VOC. (a): cosine and step schedules for batch size 64. (b): Validation mAP comparison curves using step and cosine learning scheduler.

3.5. Synchronized Batch Normalization

In recent years, the massive computation requirements forced training environments to equip multiple devices (usually GPUs) to accelerate training. Despite handling different hyper-parameters in response to larger batch sizes during training, Batch Normalization [10] is drawing the attention of multi-device users due to the implementation details. Although the typical implementation of Batch Normalization working on multiple devices (GPUs) is fast (with no communication overhead), it inevitably reduces the size of batch size and causing slightly different statistics during computation, which potentially degraded performance. This is not a significant issue in some standard vision tasks such as ImageNet classification (as the batch size per device is usually large enough to obtain good statistics). However, it will hurt the performance in some tasks that the batch size is usually very small (*e.g*., 1 per GPU). Recently, Peng *et al*. has proved the importance of synchronized batch normalization in object detection with the analysis made by Pen *et al*. [14]. In this work, we review the importance of Synchronized Batch Normalization with YOLOv3 [16] to evaluate the impacts of relatively smaller batch-size on each GPU as training image shape is significantly larger than image classification tasks.

3.6. Random shapes training for single-stage object detection networks

Natural training images are of various shapes. To fit memory limitation and allow simpler batching, many single-stage object detection networks are trained with fixed shapes [12, 15]. To reduce risk of overfitting

and improve generalization of network predictions, We follow the approach of random shapes training as in Redmon *et al*. [16]. More specifically, a mini-batch of N training images is resized to $N \times 3 \times H \times W$, where H and W are multipliers of common divisor $D = randint(1, k)$. For example, we use $H = W \in \{320, 352, 384, 416, 448, 480, 512, 544, 576, 608\}$ for YOLOv3 training.

4. Experiments

Model	mAP	Delta
- data augmentation	64.26	-15.99
baseline	80.25	0
+ synchronize BN	80.81	+0.56
+ random training shapes	81.23	+0.98
+ cosine lr schedule	81.69	+1.44
+ class label smoothing	82.14	+1.89
+ mixup	**83.68**	**+3.43**

Table 2. Training Refinements on YOLOv3, evaluated at 416×416 on Pascal VOC 2007 test set.

Model	mAP	Delta
- data augmentation	77.61	-0.16
baseline	77.77	0
+ cosine lr schedule	79.59	+1.82
+ class label smoothing	80.23	+2.46
+ mixup	**81.32**	**+3.55**

Table 3. Training Refinements on Faster-RCNN, evaluated at 600×1000 on Pascal VOC 2007 test set.

In order to compare incremental improvements of all tweaks for object detection, we used YOLOv3 [16] and Faster-RCNN [17] as representatives for single and multiple stages pipelines, respectively. To accommodate massive training tasks, we use Pascal VOC [2] for fine-graded trick evaluation, and COCO [11] dataset for overall performance gain and generalization ability verification.

Pascal VOC. Following the common settings used in Fast-RCNN [3] and SSD [12], we use Pascal VOC 2007 *trainval* and 2012 *trainval* for training and 2007 test set for validation. The results are reported in mean average precision defined in Pascal VOC development kit [2]. For YOLOv3 models, we consistently measure mean average precision (mAP) at 416×416 resolution. If random shape training is enabled, YOLOv3 models will be fed with random resolutions from 320×320 to 608×608 with 32×32 increments, otherwise they are always trained with fixed 416×416 input data. Faster RCNN models take arbitrary input resolutions. In order to regulate training memory consumption, the shorter sides of input images are resized to 600 pixels while ensuring the longer side in smaller than 1000 pixels. Training and validation of Faster-RCNN models follow the same pre-processing steps, except that training images have chances of 0.5 to flip horizontally as additional data augmentation. The incremental evaluations of YOLOv3 and Faster-RCNN with our bags of freebies (BoF) are detailed in Table. 2 and Table. 3, respectively. The results are motivating that by stacking the tricks, YOLOv3 models can achieve up to 3.5% performance gain and Faster-RCNN models have less effective tricks but obtained similar performance bump. One phenomena has been discovered that data augmentation has minimal effect on multi-stage pipelines, *i.e*., Faster-RCNN, but poses significant compact to YOLOv3 models, due to the lack of spatial mutational operations.

MS COCO. COCO 2017 is 10 times larger than Pascal VOC and contains tiny objects compared with PASCAL VOC. We use MS COCO to validate the generalization of our bags of tricks in this work. We use similar training and validation settings as Pascal VOC, except that Faster-RCNN models are resized to 800×1300 pixels in response to smaller objects. The results are shown in Table. 4. In summary, our proposed bags of freebies boosted Faster-RCNN models by 1.1% and 1.7% absolute mean AP over existing state-of-the-art implementations [5] with ResNet 50 and 101 base models, respectively. Our proposed bag of tricks also outperforms existing YOLOv3 models by as large as 4.0% absolute mAP. Note that all these results are obtained by generating better weights in a fully compatible inference model, *i.e*., all these achievements are free lunch during inference.

Impact of mixup on different phases of training detection network Mixup can be applied in two phases of object detection networks: 1) pre-training classification network backbone with mixup [8, 23]; 2) training detection networks using proposed visually coherent image mixup for object detection. We compare the results using Darknet 53-layer based YOLO3 [16] implementation and ResNet101 [7] based Faster-RCNN [17] in Table. 5 and Table. 6, respectively. While the results proved the consistent improvements by adopting mixup to either training phases, it is also notable that applying mixup in both phases can produce more significant gains as $1 + 1 > 2$.

5. Conclusion

In this paper, we proposed a bag of training enhancements significantly improved model performances while introducing zero overhead to the inference environment. Our empirical experiments of YOLOv3 [16] and Faster-RCNN [17] on Pascal VOC and COCO datasets show that the bag of tricks are consistently improving object detection models. By stacking all these tweaks, we are observing no signs of degradation of any level and suggest a wider adoption to

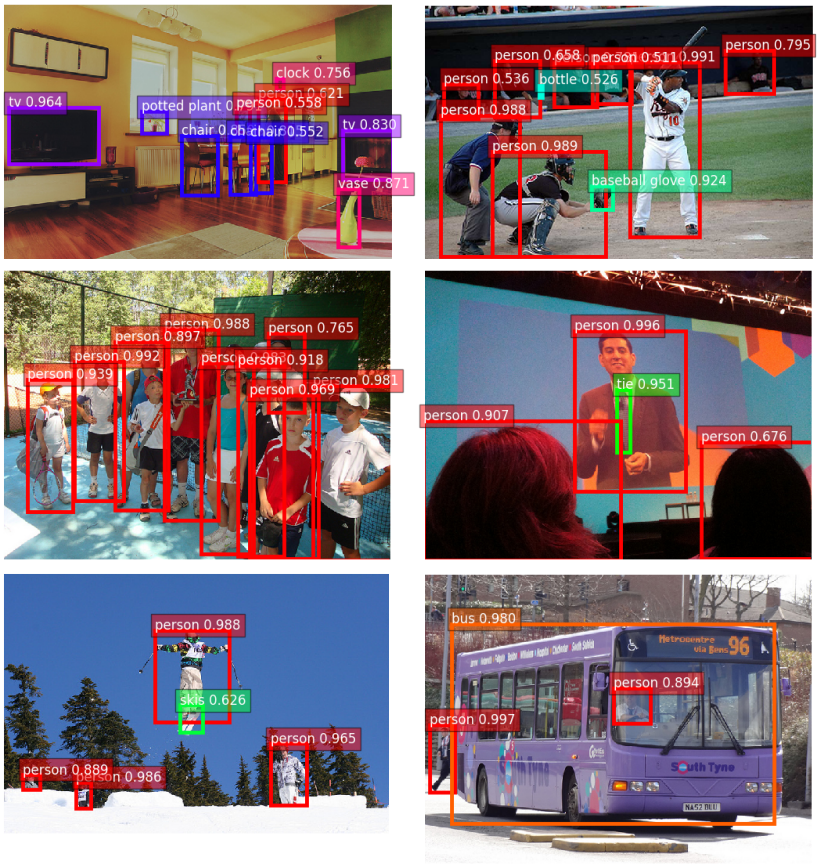

Figure 7. Example detection results by applying BoF on COCO 2017 validation set.

Model	Orig $AP^{0.5:0.95}_{bbox}$	**Our BoF** $AP^{0.5:0.95}_{bbox}$	Absolute delta
Faster-RCNN R50 [5]	36.5	**37.6**	**+1.1**
Faster-RCNN R101 [5]	39.4	**41.1**	**+1.7**
YOLOv3 @320 [16]	28.2	**33.6**	**+5.4**
YOLOv3 @416 [16]	31.0	**36.0**	**+5.0**
YOLOv3 @608 [16]	33.0	**37.0**	**+4.0**

Table 4. Overview of improvements achieved by applying bag of freebies(BoF), evaluated on MS COCO [11] 2017 val set. Note that YOLOv3 models can be evaluated at different input resolutions with same weights, our BoF improves evaluation results more significantly at lower resolution levels.

	-Mixup YOLO3	+Mixup YOLO3
-Mixup darknet53	35.0	35.3
+Mixup darknet53	36.4	37.0

Table 5. Combined analysis of impacts of mix-up methodology for pre-trained image classification and detection network.

	-Mixup FRCNN	+Mixup FRCNN
-Mixup R101	39.9	40.1
+Mixup R101	40.1	41.1

Table 6. Combined analysis of impacts of mix-up methodology for pre-trained image classification and detection network.

future object detection training pipelines. All existing and future work will be included as part of open source GluonCV repository [1].

References

[1] DMLC. Gluoncv toolkit. `https://github.com/dmlc/gluon-cv`, 2018.

[2] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and

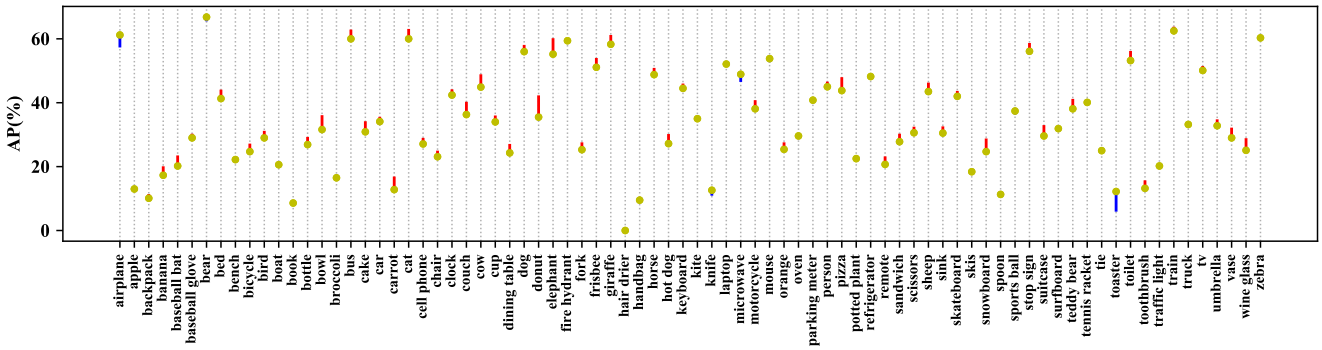

Figure 8. **COCO 80 category AP analysis with YOLOv3 [16].** Red lines indicate performance gain using BoF, while blue lines indicate performance drop.

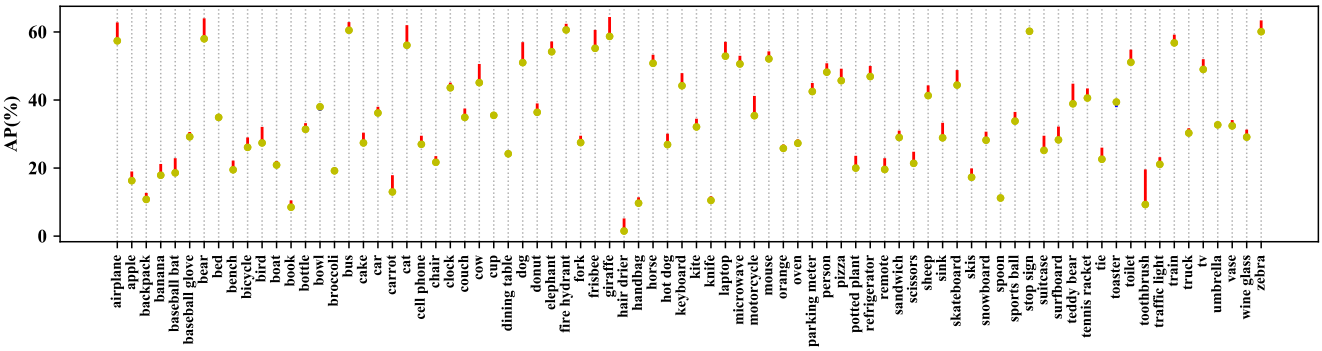

Figure 9. **COCO 80 category AP analysis with Faster-RCNN resnet50 [17].** Red lines indicate performance gain using BoF, while blue lines indicate performance drop.

A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[3] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[5] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[6] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[7] 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.

[8] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. *arXiv preprint arXiv:1812.01187*, 2018.

[9] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[12] 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.

[13] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[14] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun. Megdet: A large mini-batch object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6181–6189, 2018.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[16] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[17] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[18] A. Rosenfeld, R. Zemel, and J. K. Tsotsos. The elephant in the room. *arXiv preprint arXiv:1808.03305*, 2018.

[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[22] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[23] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.