

# Frustratingly Simple Few-Shot Object Detection

Xin Wang<sup>\*1</sup> Thomas E. Huang<sup>\*2</sup> Trevor Darrell<sup>1</sup> Joseph E. Gonzalez<sup>1</sup> Fisher Yu<sup>1</sup>

## Abstract

Detecting rare objects from a few examples is an emerging problem. Prior works show meta-learning is a promising approach. But, fine-tuning techniques have drawn scant attention. We find that fine-tuning only the last layer of existing detectors on rare classes is crucial to the few-shot object detection task. Such a simple approach outperforms the meta-learning methods by roughly 2~20 points on current benchmarks and sometimes even doubles the accuracy of the prior methods. However, the high variance in the few samples often leads to the unreliability of existing benchmarks. We revise the evaluation protocols by sampling multiple groups of training examples to obtain stable comparisons and build new benchmarks based on three datasets: PASCAL VOC, COCO and LVIS. Again, our fine-tuning approach establishes a new state of the art on the revised benchmarks. The code as well as the pretrained models are available at <https://github.com/ucbdrive/few-shot-object-detection>.

## 1. Introduction

Machine perception systems have witnessed significant progress in the past years. Yet, our ability to train models that generalize to novel concepts without abundant labeled data is still far from satisfactory when compared to human visual systems. Even a toddler can easily recognize a new concept with very little instruction (Landau et al., 1988; Samuelson & Smith, 2005; Smith et al., 2002).

The ability to generalize from only a few examples (so called few-shot learning) has become a key area of interest in the machine learning community. Many (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017; Hariharan & Girshick, 2017; Gidaris & Komodakis, 2018; Wang et al., 2019a) have explored techniques to transfer knowledge from the

<sup>\*</sup>Equal contribution <sup>1</sup>EECS, UC Berkeley <sup>2</sup>EECS, University of Michigan. Correspondence to: Xin Wang <xinw@berkeley.edu>, Thomas E. Huang <thomaseh@umich.edu>.

data-abundant base classes to the data-scarce novel classes through *meta-learning*. They use simulated few-shot tasks by sampling from base classes during training to learn to learn from the few examples in the novel classes.

However, much of this work has focused on basic image classification tasks. In contrast, few-shot object detection has received far less attention. Unlike image classification, object detection requires the model to not only recognize the object types but also localize the targets among millions of potential regions. This additional subtask substantially raises the overall complexity. Several (Kang et al., 2019; Yan et al., 2019; Wang et al., 2019b) have attempted to tackle the under-explored few-shot object detection task, where only a few labeled bounding boxes are available for novel classes. These methods attach meta learners to existing object detection networks, following the meta-learning methods for classification. But, current evaluation protocols suffer from statistical unreliability, and the accuracy of baseline methods, especially simple fine-tuning, on few-object detection are not consistent in the literature.

In this work, we propose improved methods to evaluate few-shot object detection. We carefully examine fine-tuning based approaches, which are considered to be underperforming in the previous works (Kang et al., 2019; Yan et al., 2019; Wang et al., 2019b). We focus on the training schedule and the instance-level feature normalization of the object detectors in model design and training based on fine-tuning.

We adopt a two-stage training scheme for fine-tuning as shown in Figure 1. We first train the entire object detector, such as Faster R-CNN (Ren et al., 2015), on the data-abundant base classes, and then only fine-tune the last layers of the detector on a small balanced training set consisting of both base and novel classes while freezing the other parameters of the model. During the fine-tuning stage, we introduce instance-level feature normalization to the box classifier inspired by Gidaris & Komodakis (2018); Qi et al. (2018); Chen et al. (2019).

We find that this two-stage fine-tuning approach (TFA) outperforms all previous state-of-the-art meta-learning based methods by 2~20 points on the existing PASCAL VOC (Everingham et al., 2007) and COCO (Lin et al., 2014) benchmarks. When training on a single novel example (one-shot

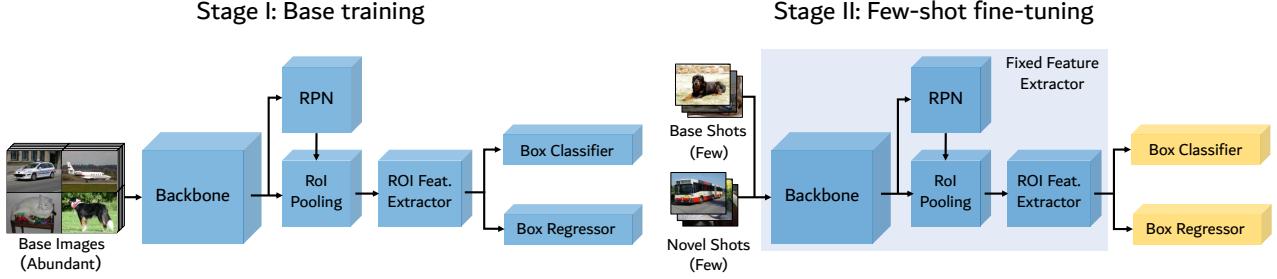


Figure 1. Illustration of our two-stage fine-tuning approach (TFA). In the base training stage, the entire object detector, including both the feature extractor  $\mathcal{F}$  and the box predictor, are jointly trained on the base classes. In the few-shot fine-tuning stage, the feature extractor components are fixed and only the box predictor is fine-tuned on a balanced subset consisting of both the base and novel classes.

learning), our method can achieve twice the accuracy of prior sophisticated state-of-the-art approaches.

Several issues with the existing evaluation protocols prevent consistent model comparisons. The accuracy measurements have high variance, making published comparisons unreliable. Also, the previous evaluations only report the detection accuracy on the novel classes, and fail to evaluate knowledge retention on the base classes.

To resolve these issues, we build new benchmarks on three datasets: PASCAL VOC, COCO and LVIS (Gupta et al., 2019). We sample different groups of few-shot training examples for multiple runs of the experiments to obtain a stable accuracy estimation and quantitatively analyze the variances of different evaluation metrics. The new evaluation reports the average precision (AP) on both the base classes and novel classes as well as the mean AP on all classes, referred to as the generalized few-shot learning setting in the few-shot classification literature (Hariharan & Girshick, 2017; Wang et al., 2019a).

Our fine-tuning approach establishes new states of the art on the benchmarks. On the challenging LVIS dataset, our two-stage training scheme improves the average detection precision of rare classes ( $< 10$  images) by  $\sim 4$  points and common classes (10-100 images) by  $\sim 2$  points with negligible precision loss for the frequent classes ( $> 100$  images).

## 2. Related Work

Our work is related to the rich literature on few-shot image classification, which uses various meta-learning based or metric-learning based methods. We also draw connections between our work and the existing meta-learning based few-shot object detection methods. To the best of our knowledge, we are the first to conduct a systematic analysis of fine-tuning based approaches on few-shot object detection.

**Meta-learning.** The goal of meta-learning is to acquire task-level meta knowledge that can help the model quickly adapt to new tasks and environments with very few labeled

examples. Some (Finn et al., 2017; Rusu et al., 2018; Nichol et al., 2018) learn to fine-tune and aim to obtain a good parameter initialization that can adapt to new tasks with a few stochastic gradient updates. Another popular line of research on meta-learning is to use parameter generation during adaptation to novel tasks. Gidaris & Komodakis (2018) propose an attention-based weight generator to generate the classifier weights for the novel classes. Wang et al. (2019a) construct task-aware feature embeddings by generating parameters for the feature layers. These approaches have only been used for few-shot image classification and not on more challenging tasks like object detection.

However, some (Chen et al., 2019) raise concerns about the reliability of the results given that a consistent comparison of different approaches is missing. Some simple fine-tuning based approaches, which draw little attention in the community, turn out to be more favorable than many prior works that use meta-learning on few-shot image classification (Chen et al., 2019; Dhillon et al., 2019). As for the emerging few-shot object detection task, there is neither consensus on the evaluation benchmarks nor a consistent comparison of different approaches due to the increased network complexity, obscure implementation details, and variances in evaluation protocols.

**Metric-learning.** Another line of work (Koch, 2015; Snell et al., 2017; Vinyals et al., 2016) focuses on learning to compare or metric-learning. Intuitively, if the model can construct distance metrics to estimate the similarity between two input images, it may generalize to novel categories with few labeled instances. More recently, several (Chen et al., 2019; Gidaris & Komodakis, 2018; Qi et al., 2018) adopt a cosine similarity based classifier to reduce the intra-class variance on the few-shot classification task, which leads to favorable performance compared to many meta-learning based approaches. Our method also adopts a cosine similarity classifier to classify the categories of the region proposals. However, we focus on the instance-level distance measurement rather than on the image level.

**Few-shot object detection.** There are several early attempts at few-shot object detection using meta-learning. Kang et al. (2019) and Yan et al. (2019) apply feature re-weighting schemes to a single-stage object detector (YOLOv2) and a two-stage object detector (Faster R-CNN), with the help of a meta learner that takes the support images (*i.e.*, a small number of labeled images of the novel/base classes) as well as the bounding box annotations as inputs. Wang et al. (2019b) propose a weight prediction meta-model to predict parameters of category-specific components from the few examples while learning the category-agnostic components from base class examples.

In all these works, fine-tuning based approaches are considered as baselines with worse performance than meta-learning based approaches. They consider *jointly fine-tuning*, where base classes and novel classes are trained together, and *fine-tuning the entire model*, where the detector is first trained on the base classes only and then fine-tuned on a balanced set with both base and novel classes. In contrast, we find that fine-tuning only the last layer of the object detector on the balanced subset and keeping the rest of model fixed can substantially improve the detection accuracy, outperforming all the prior meta-learning based approaches. This indicates that feature representations learned from the base classes might be able to transfer to the novel classes and simple adjustments to the box predictor can provide strong performance gain (Dhillon et al., 2019).

### 3. Algorithms for Few-Shot Object Detection

In this section, we start with the preliminaries on the few-shot object detection setting. Then, we talk about our two-stage fine-tuning approach in Section 3.1. Section 3.2 summarizes the previous meta-learning approaches.

We follow the few-shot object detection settings introduced in Kang et al. (2019). There are a set of base classes  $C_b$  that have many instances and a set of novel classes  $C_n$  that have only  $K$  (usually less than 10) instances per category. For an object detection dataset  $\mathcal{D} = \{(x, y), x \in \mathcal{X}, y \in \mathcal{Y}\}$ , where  $x$  is the input image and  $y = \{(c_i, l_i), i = 1, \dots, N\}$  denotes the categories  $c \in C_b \cup C_n$  and bounding box coordinates  $l$  of the  $N$  object instances in the image  $x$ . For synthetic few-shot datasets using PASCAL VOC and COCO, the novel set for training is balanced and each class has the same number of annotated objects (*i.e.*,  $K$ -shot). The recent LVIS dataset has a natural long-tail distribution, which does not have the manual  $K$ -shot split. The classes in LVIS are divided into *frequent* classes (appearing in more than 100 images), *common* classes (10-100 images), and *rare* classes (less than 10 images). We consider both synthetic and natural datasets in our work and follow the naming convention of  $k$ -shot for simplicity.

The few-shot object detector is evaluated on a test set of both

the base classes and the novel classes. The goal is to optimize the detection accuracy measured by average precision (AP) of the novel classes as well as the base classes. This setting is different from the  $N$ -way- $K$ -shot setting (Finn et al., 2017; Vinyals et al., 2016; Snell et al., 2017) commonly used in few-shot classification.

#### 3.1. Two-stage fine-tuning approach

We describe our two-stage fine-tuning approach (TFA) for few-shot object detection in this section. We adopt the widely used Faster R-CNN (Ren et al., 2015), a two-stage object detector, as our base detection model. As shown in Figure 1, the feature learning components, referred to as  $\mathcal{F}$ , of a Faster R-CNN model include the backbone (*e.g.*, ResNet (He et al., 2016), VGG16 (Simonyan & Zisserman, 2014)), the region proposal network (RPN), as well as a two-layer fully-connected (FC) sub-network as a proposal-level feature extractor. There is also a box predictor composed of a box classifier  $\mathcal{C}$  to classify the object categories and a box regressor  $\mathcal{R}$  to predict the bounding box coordinates. Intuitively, the backbone features as well as the RPN features are class-agnostic. Therefore, features learned from the base classes are likely to transfer to the novel classes without further parameter updates. The key component of our method is to separate the feature representation learning and the box predictor learning into two stages.

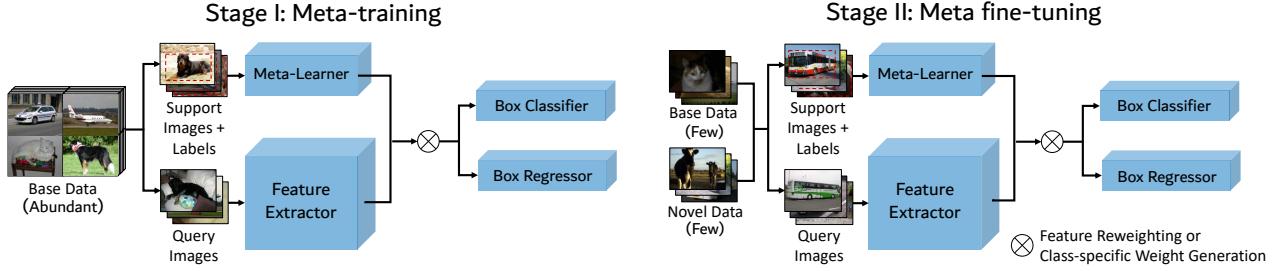
**Base model training.** In the first stage, we train the feature extractor and the box predictor only on the base classes  $C_b$ , with the same loss function used in Ren et al. (2015). The joint loss is,

$$\mathcal{L} = \mathcal{L}_{\text{rpn}} + \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{loc}}, \quad (1)$$

where  $\mathcal{L}_{\text{rpn}}$  is applied to the output of the RPN to distinguish foreground from backgrounds and refine the anchors,  $\mathcal{L}_{\text{cls}}$  is a cross-entropy loss for the box classifier  $\mathcal{C}$ , and  $\mathcal{L}_{\text{loc}}$  is a smoothed  $L_1$  loss for the box regressor  $\mathcal{R}$ .

**Few-shot fine-tuning.** In the second stage, we create a small balanced training set with  $K$  shots per class, containing both base and novel classes. We assign randomly initialized weights to the box prediction networks for the novel classes and fine-tune only the box classification and regression networks, namely the last layers of the detection model, while keeping the entire feature extractor  $\mathcal{F}$  fixed. We use the same loss function in Equation 1 and a smaller learning rate. The learning rate is reduced by 20 from the first stage in all our experiments.

**Cosine similarity for box classifier.** We consider using a classifier based on cosine similarity in the second fine-tuning stage, inspired by Gidaris & Komodakis (2018); Qi et al. (2018); Chen et al. (2019). The weight matrix  $W \in \mathbb{R}^{d \times c}$  of the box classifier  $\mathcal{C}$  can be written as  $[w_1, w_2, \dots, w_c]$ , where  $w_c \in \mathbb{R}^d$  is the per-class weight vector. The output of  $\mathcal{C}$  is



**Figure 2.** Abstraction of the meta-learning based few-shot object detectors. A meta-learner is introduced to acquire task-level meta information and help the model generalize to novel classes through feature re-weighting (e.g., FSRW and Meta R-CNN) or weight generation (e.g., MetaDet). A two-stage training approach (meta-training and meta fine-tuning) with episodic learning is commonly adopted.

scaled similarity scores  $S$  of the input feature  $\mathcal{F}(x)$  and the weight vectors of different classes. The entries in  $S$  are

$$s_{i,j} = \frac{\alpha \mathcal{F}(x)_i^\top w_j}{\|\mathcal{F}(x)_i\| \|w_j\|}, \quad (2)$$

where  $s_{i,j}$  is the similarity score between the  $i$ -th object proposal of the input  $x$  and the weight vector of class  $j$ .  $\alpha$  is the scaling factor. We use a fixed  $\alpha$  of 20 in our experiments. We find empirically that the instance-level feature normalization used in the cosine similarity based classifier helps reduce the intra-class variance and improves the detection accuracy of novel classes with less decrease in detection accuracy of base classes when compared to a FC-based classifier, especially when the number of training examples is small.

### 3.2. Meta-learning based approaches

We describe the existing meta-learning based few-shot object detection networks, including FSRW (Kang et al., 2019), Meta R-CNN (Yan et al., 2019) and MetaDet (Wang et al., 2019b), in this section to draw comparisons with our approach. Figure 2 illustrates the structures of these networks. In meta-learning approaches, in addition to the base object detection model that is either single-stage or two-stage, a meta-learner is introduced to acquire class-level meta knowledge and help the model generalize to novel classes through feature re-weighting, such as FSRW and Meta R-CNN, or class-specific weight generation, such as MetaDet. The input to the meta learner is a small set of support images with the bounding box annotations of the target objects.

The base object detector and the meta-learner are often jointly trained using episodic training (Vinyals et al., 2016). Each episode is composed of a supporting set of  $N$  objects and a set of query images. In FSRW and Meta R-CNN, the support images and the binary masks of the annotated objects are used as input to the meta-learner, which generates class reweighting vectors that modulate the feature representation of the query images. As shown in Figure 2,

the training procedure is also split into a meta-training stage, where the model is only trained on the data of the base classes, and a meta fine-tuning stage, where the support set includes the few examples of the novel classes and a subset of examples from the base classes.

Both the meta-learning approaches and our approach have a two-stage training scheme. However, we find that the episodic learning used in meta-learning approaches can be very memory inefficient as the number of classes in the supporting set increases. Our fine-tuning method only fine-tunes the last layers of the network with a normal batch training scheme, which is much more memory efficient.

## 4. Experiments

In this section, we conduct extensive comparisons with previous methods on the existing few-shot object detection benchmarks using PASCAL VOC and COCO, where our approach can obtain about 2~20 points improvement in all settings (Section 4.1). We then introduce a new benchmark on three datasets (PASCAL VOC, COCO and LVIS) with revised evaluation protocols to address the unreliability of previous benchmarks (Section 4.2). We also provide various ablation studies and visualizations in Section 4.3.

**Implementation details.** We use Faster R-CNN (Ren et al., 2015) as our base detector and Resnet-101 (He et al., 2016) with a Feature Pyramid Network (Lin et al., 2017) as the backbone. All models are trained using SGD with a mini-batch size of 16, momentum of 0.9, and weight decay of 0.0001. A learning rate of 0.02 is used during base training and 0.001 during few-shot fine-tuning. For more details, the code is available at <https://github.com/ucbdrive/few-shot-object-detection>.

### 4.1. Existing few-shot object detection benchmark

**Existing benchmarks.** Following the previous work (Kang et al., 2019; Yan et al., 2019; Wang et al., 2019b), we first

**Table 1.** Few-shot detection performance (mAP50) on the PASCAL VOC dataset. We evaluate the performance on three different sets of novel classes. Our approach consistently outperforms baseline methods by a large margin (about 2~20 points), especially when the number of shots is low. FRCN stands for Faster R-CNN. TFA w/ cos is our approach with a cosine similarity based box classifier.

| Method / Shot                      | Backbone     | Novel Set 1 |             |             |             |             | Novel Set 2 |             |             |             |             | Novel Set 3 |             |             |             |             |
|------------------------------------|--------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|                                    |              | 1           | 2           | 3           | 5           | 10          | 1           | 2           | 3           | 5           | 10          | 1           | 2           | 3           | 5           | 10          |
| YOLO-joint (Kang et al., 2019)     | YOLOv2       | 0.0         | 0.0         | 1.8         | 1.8         | 1.8         | 0.0         | 0.1         | 0.0         | 1.8         | 0.0         | 1.8         | 1.8         | 1.8         | 3.6         | 3.9         |
| YOLO-ft (Kang et al., 2019)        |              | 3.2         | 6.5         | 6.4         | 7.5         | 12.3        | 8.2         | 3.8         | 3.5         | 3.5         | 7.8         | 8.1         | 7.4         | 7.6         | 9.5         | 10.5        |
| YOLO-ft-full (Kang et al., 2019)   |              | 6.6         | 10.7        | 12.5        | 24.8        | 38.6        | 12.5        | 4.2         | 11.6        | 16.1        | 33.9        | 13.0        | 15.9        | 15.0        | 32.2        | 38.4        |
| FSRW (Kang et al., 2019)           |              | 14.8        | 15.5        | 26.7        | 33.9        | 47.2        | 15.7        | 15.3        | 22.7        | 30.1        | 40.5        | 21.3        | 25.6        | 28.4        | 42.8        | 45.9        |
| MetaDet (Wang et al., 2019b)       |              | 17.1        | 19.1        | 28.9        | 35.0        | 48.8        | 18.2        | 20.6        | 25.9        | 30.6        | 41.5        | 20.1        | 22.3        | 27.9        | 41.9        | 42.9        |
| FRCN+joint (Wang et al., 2019b)    | FRCN w/VGG16 | 0.3         | 0.0         | 1.2         | 0.9         | 1.7         | 0.0         | 0.0         | 1.1         | 1.9         | 1.7         | 0.2         | 0.5         | 1.2         | 1.9         | 2.8         |
| FRCN+joint-ft (Wang et al., 2019b) |              | 9.1         | 10.9        | 13.7        | 25.0        | 39.5        | 10.9        | 13.2        | 17.6        | 19.5        | 36.5        | 15.0        | 15.1        | 18.3        | 33.1        | 35.9        |
| MetaDet (Wang et al., 2019b)       |              | 18.9        | 20.6        | 30.2        | 36.8        | 49.6        | 21.8        | 23.1        | 27.8        | 31.7        | 43.0        | 20.6        | 23.9        | 29.4        | 43.9        | 44.1        |
| FRCN+joint (Yan et al., 2019)      | FRCN w/R-101 | 2.7         | 3.1         | 4.3         | 11.8        | 29.0        | 1.9         | 2.6         | 8.1         | 9.9         | 12.6        | 5.2         | 7.5         | 6.4         | 6.4         | 6.4         |
| FRCN+ft (Yan et al., 2019)         |              | 11.9        | 16.4        | 29.0        | 36.9        | 36.9        | 5.9         | 8.5         | 23.4        | 29.1        | 28.8        | 5.0         | 9.6         | 18.1        | 30.8        | 43.4        |
| FRCN+ft-full (Yan et al., 2019)    |              | 13.8        | 19.6        | 32.8        | 41.5        | 45.6        | 7.9         | 15.3        | 26.2        | 31.6        | 39.1        | 9.8         | 11.3        | 19.1        | 35.0        | 45.1        |
| Meta R-CNN (Yan et al., 2019)      |              | 19.9        | 25.5        | 35.0        | 45.7        | 51.5        | 10.4        | 19.4        | 29.6        | 34.8        | <b>45.4</b> | 14.3        | 18.2        | 27.5        | 41.2        | 48.1        |
| FRCN+ft-full (Our Impl.)           | FRCN w/R-101 | 15.2        | 20.3        | 29.0        | 40.1        | 45.5        | 13.4        | 20.6        | 28.6        | 32.4        | 38.8        | 19.6        | 20.8        | 28.7        | 42.2        | 42.1        |
| TFA w/ fc (Ours)                   |              | 36.8        | 29.1        | 43.6        | <b>55.7</b> | <b>57.0</b> | 18.2        | <b>29.0</b> | 33.4        | <b>35.5</b> | 39.0        | 27.7        | 33.6        | 42.5        | 48.7        | <b>50.2</b> |
| TFA w/ cos (Ours)                  |              | <b>39.8</b> | <b>36.1</b> | <b>44.7</b> | <b>55.7</b> | 56.0        | <b>23.5</b> | 26.9        | <b>34.1</b> | 35.1        | 39.1        | <b>30.8</b> | <b>34.8</b> | <b>42.8</b> | <b>49.5</b> | 49.8        |

evaluate our approach on PASCAL VOC 2007+2012 and COCO, using the same data splits and training examples provided by Kang et al. (2019). For the few-shot PASCAL VOC dataset, the 20 classes are randomly divided into 15 base classes and 5 novel classes, where the novel classes have  $K = 1, 2, 3, 5, 10$  objects per class sampled from the combination of the trainval sets of the 2007 and 2012 versions for training. Three random split groups are considered in this work. PASCAL VOC 2007 test set is used for evaluation. For COCO, the 60 categories disjoint with PASCAL VOC are used as base classes while the remaining 20 classes are used as novel classes with  $K = 10, 30$ . For evaluation metrics, AP50 (matching threshold is 0.5) of the novel classes is used on PASCAL VOC and the COCO-style AP of the novel classes is used on COCO.

**Baselines.** We compare our approach with the meta-learning approaches FSRW, Meta-RCNN and MetaDet together with the fine-tuning based approaches: *jointly training*, denoted by FRCN/YOLO+joint, where the base and novel class examples are jointly trained in one stage, and *fine-tuning the entire model*, denoted by FRCN/YOLO+ft+full, where both the feature extractor  $\mathcal{F}$  and the box predictor ( $\mathcal{C}$  and  $\mathcal{R}$ ) are jointly fine-tuned until convergence in the second fine-tuning stage. FRCN is Faster R-CNN for short. Fine-tuning with less iterations, denoted by FRCN/YOLO+ft, are reported in Kang et al. (2019) and Yan et al. (2019).

**Results on PASCAL VOC.** We provide the average AP50 of the novel classes on PASCAL VOC with three random splits in Table 1. Our approach uses ResNet-101 as the backbone, similar to Meta R-CNN. We implement FRCN+ft+full in our framework, which roughly matches the results reported in Yan et al. (2019). MetaDet

**Table 2.** Few-shot detection performance for the base and novel classes on Novel Set 1 of the PASCAL VOC dataset. Our approach outperforms baselines on both base and novel classes and does not degrade the performance on the base classes greatly.

| # shots | Method                          | Base AP50   | Novel AP50  |
|---------|---------------------------------|-------------|-------------|
| 3       | FRCN+ft-full (Yan et al., 2019) | 63.6        | 32.8        |
|         | Meta R-CNN (Yan et al., 2019)   | 64.8        | 35.0        |
|         | Train base only (Our Impl.)     | <b>80.8</b> | 9.0         |
|         | FRCN+ft-full (Our Impl.)        | 66.1        | 29.0        |
|         | TFA w/ cos (Ours)               | 79.1        | <b>44.7</b> |
| 10      | FRCN+ft-full (Yan et al., 2019) | 61.3        | 45.6        |
|         | Meta R-CNN (Yan et al., 2019)   | 67.9        | 51.5        |
|         | Train base only                 | <b>80.8</b> | 9.0         |
|         | FRCN+ft-full (Our Impl.)        | 66.0        | 45.5        |
|         | TFA w/ cos (Ours)               | 78.4        | <b>56.0</b> |

uses VGG16 as the backbone, but the performance is similar and sometimes worse compared to Meta R-CNN.

In all different data splits and different numbers of training shots, our approach (the last row) is able to outperform the previous methods by a large margin. It even doubles the performance of the previous approaches in the one-shot cases. The improvements, up to 20 points, is much larger than the gap among the previous meta-learning based approaches, indicating the effectiveness of our approach. We also compare the cosine similarity based box classifier (TFA +w/ cos) with a normal FC-based classifier (TFA +w/ fc) and find that TFA +w/ cos is better than TFA +w/ fc on extremely low shots (e.g., 1-shot), but the two are roughly similar when there are more training shots, e.g., 10-shot.

For more detailed comparisons, we cite the numbers from Yan et al. (2019) of their model performance on the base classes in Table 2. We find that our model has a

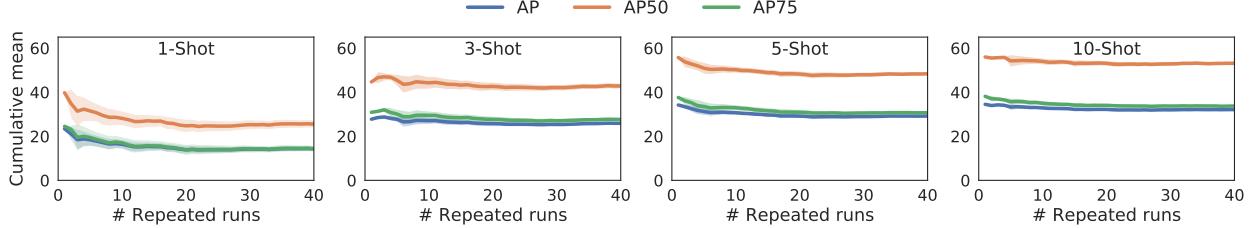


Figure 3. Cumulative means with 95% confidence intervals across 40 repeated runs, computed on the novel classes of the first split of PASCAL VOC. The means and variances become stable after around 30 runs.

much higher average AP on the base classes than Meta R-CNN with a gap of about 10 to 15 points. To eliminate the differences in implementation details, we also report our re-implementation of FRCN+ft-full and training base only, where the base classes should have the highest accuracy as the model is only trained on the base classes examples. We find that our model has a small decrease in performance, less than 2 points, on the base classes.

Table 3. Few-shot detection performance for the novel categories on the COCO dataset. Our approach consistently outperforms baseline methods across all shots and metrics.

| Model                           | novel AP    |             | novel AP75 |             |
|---------------------------------|-------------|-------------|------------|-------------|
|                                 | 10          | 30          | 10         | 30          |
| FSRW (Kang et al., 2019)        | 5.6         | 9.1         | 4.6        | 7.6         |
| MetaDet (Wang et al., 2019b)    | 7.1         | 11.3        | 6.1        | 8.1         |
| FRCN+ft+full (Yan et al., 2019) | 6.5         | 11.1        | 5.9        | 10.3        |
| Meta R-CNN (Yan et al., 2019)   | 8.7         | 12.4        | 6.6        | 10.8        |
| FRCN+ft-full (Our Impl.)        | 9.2         | 12.5        | 9.2        | 12.0        |
| TFA w/ fc (Ours)                | <b>10.0</b> | 13.4        | 9.2        | 13.2        |
| TFA w/ cos (Ours)               | <b>10.0</b> | <b>13.7</b> | <b>9.3</b> | <b>13.4</b> |

**Results on COCO.** Similarly, we report the average AP and AP75 of the 20 novel classes on COCO in Table 3. AP75 means matching threshold is 0.75, a more strict metric than AP50. Again, we consistently outperform previous methods across all shots on both novel AP and novel AP75. We achieve around 1 point improvement in AP over the best performing baseline and around 2.5 points improvement in AP75.

## 4.2. Generalized few-shot object detection benchmark

**Revised evaluation protocols.** We find several issues with existing benchmarks. First, previous evaluation protocols focus only on the performance on novel classes. This ignores the potential performance drop in base classes and thus the overall performance of the network. Second, the sample variance is large due to the few samples that are used for training. This makes it difficult to draw conclusions from comparisons against other methods, as differences in performance could be insignificant.

To address these issues, we first revise the evaluation protocol to include evaluation on base classes. On our benchmark,

we report AP on base classes (bAP) and the overall AP in addition to AP on the novel classes (nAP). This allows us to observe trends in performance on both base and novel classes, and the overall performance of the network.

Additionally, we train our models for multiple runs on different random samples of training shots to obtain averages and confidence intervals. In Figure 3, we show the cumulative mean and 95% confidence interval across 40 repeated runs with  $K = 1, 3, 5, 10$  on the first split of PASCAL VOC. Although the performance is high on the first random sample, the average decreases significantly as more samples are used. Additionally, the confidence intervals across the first few runs are large, especially in the low-shot scenario. When we use more repeated runs, the averages stabilize and the confidence intervals become small, which allows for better comparisons.

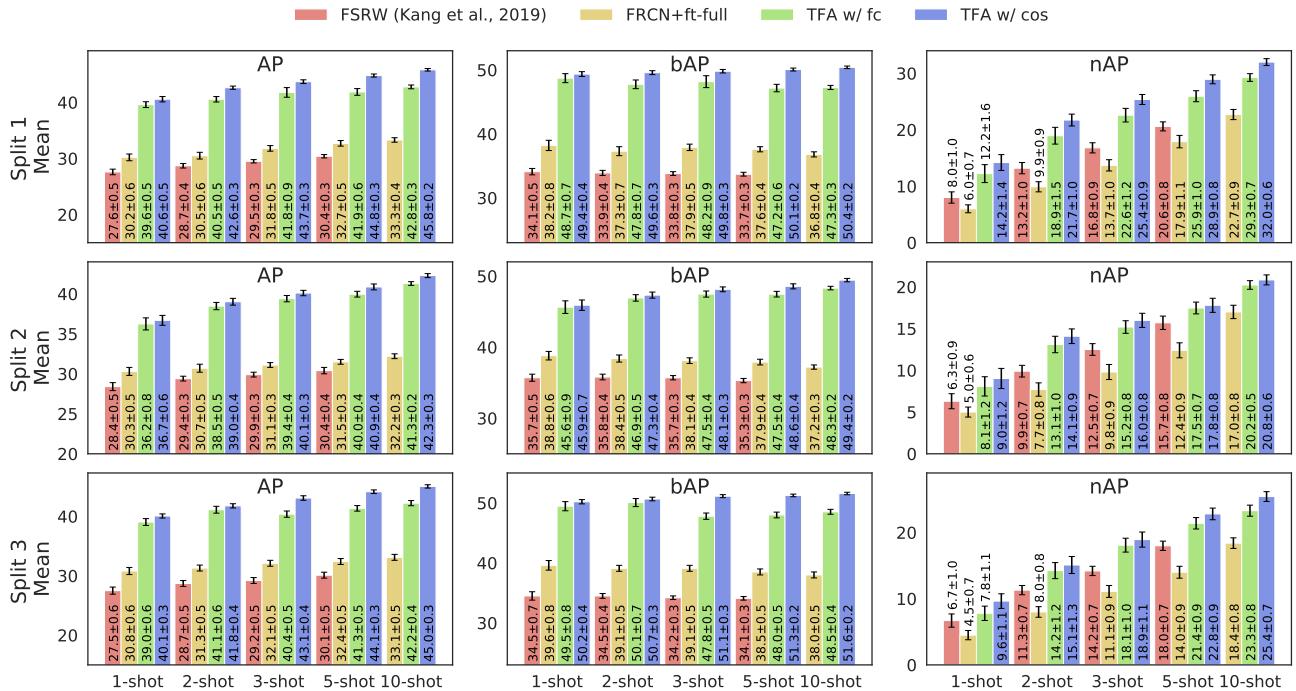
**Results on LVIS.** We evaluate our approach on the recently introduced LVIS dataset (Gupta et al., 2019). The number of images in each category in LVIS has a natural long-tail distribution. We treat the frequent and common classes as base classes, and the rare classes as novel classes. The base training is the same as before. During few-shot fine-tuning, we artificially create a balanced subset of the entire dataset by sampling up to 10 instances for each class and fine-tune on this subset.

We show evaluation results on LVIS in Table 4. Compared to the methods in Gupta et al. (2019), our approach is able to achieve better performance of  $\sim 1\text{-}1.5$  points in overall AP and  $\sim 2\text{-}4$  points in AP for rare and common classes. We also demonstrate results without using repeated sampling, which is a weighted sampling scheme that is used in Gupta et al. (2019) to address the data imbalance issue. In this setting, the baseline methods can only achieve  $\sim 2\text{-}3$  points in AP for rare classes. On the other hand, our approach is able to greatly outperform the baseline and increase the AP on rare classes by around 13 points and on common classes by around 1 point. Our two-stage fine-tuning scheme is able to address the severe data imbalance issue without needing repeated sampling.

**Results on PASCAL VOC and COCO.** We show evaluation results on generalized PASCAL VOC in Figure 4 and

**Table 4.** Generalized object detection benchmarks on LVIS. We compare our approach to the baselines provided in LVIS (Gupta et al., 2019). Our approach outperforms the corresponding baseline across all metrics, backbones, and sampling schemes.

| Method                              | Backbone      | Repeated sampling | AP          | AP50        | AP75        | APs         | APm         | API         | APr         | APc         | APf         |
|-------------------------------------|---------------|-------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Joint training (Gupta et al., 2019) | FRCN w/ R-50  |                   | 19.8        | 33.6        | 20.4        | 17.1        | 25.9        | 33.2        | 2.1         | 18.5        | <b>28.5</b> |
| TFA w/ fc (Ours)                    |               |                   | 22.3        | 37.8        | 22.0        | 18.5        | 28.2        | 36.6        | 14.3        | 21.1        | 27.0        |
| TFA w/ cos (Ours)                   |               |                   | 22.7        | 37.2        | 23.9        | 18.8        | 27.7        | 37.1        | 15.4        | 20.5        | 28.4        |
| Joint training (Gupta et al., 2019) | FRCN w/ R-50  | ✓                 | 23.1        | 38.4        | 24.3        | 18.1        | 28.3        | 36.0        | 13.0        | 22.0        | 28.4        |
| TFA w/ fc (Ours)                    |               |                   | 24.1        | 39.9        | 25.4        | 19.5        | 29.1        | 36.7        | 14.9        | 23.9        | 27.9        |
| TFA w/ cos (Ours)                   |               |                   | <b>24.4</b> | <b>40.0</b> | <b>26.1</b> | <b>19.9</b> | <b>29.5</b> | <b>38.2</b> | <b>16.9</b> | <b>24.3</b> | 27.7        |
| Joint training (Gupta et al., 2019) | FRCN w/ R-101 |                   | 21.9        | 35.8        | 23.0        | 18.8        | 28.0        | 36.2        | 3.0         | 20.8        | <b>30.8</b> |
| TFA w/ fc (Ours)                    |               |                   | 23.9        | 39.3        | 25.3        | 19.5        | 29.5        | 38.6        | 16.2        | 22.3        | 28.9        |
| TFA w/ cos (Ours)                   |               |                   | 24.3        | 39.3        | 25.8        | 20.1        | 30.2        | 39.5        | <b>18.1</b> | 21.8        | 29.8        |
| Joint training (Gupta et al., 2019) | FRCN w/ R-101 | ✓                 | 24.7        | 40.5        | 26.0        | 19.0        | 30.3        | 38.0        | 13.4        | 24.0        | 30.1        |
| TFA w/ fc (Ours)                    |               |                   | 25.4        | <b>41.8</b> | 27.0        | 19.8        | 31.1        | 39.2        | 15.5        | 26.0        | 28.6        |
| TFA w/ cos (Ours)                   |               |                   | <b>26.2</b> | <b>41.8</b> | <b>27.5</b> | <b>20.2</b> | <b>32.0</b> | <b>39.9</b> | 17.3        | <b>26.4</b> | 29.6        |



**Figure 4.** Generalized object detection benchmarks on PASCAL VOC. For each metric, we report the average and 95% confidence interval computed over 30 random samples.

COCO in Figure 5. On both datasets, we evaluate on the base classes and the novel classes and report AP scores for each. On PASCAL VOC, we evaluate our models over 30 repeated runs and report the average and the 95% confidence interval. On COCO, we provide results on 1, 2, 3, and 5 shots in addition to the 10 and 30 shots used by the existing benchmark for a better picture of performance trends in the low-shot regime. For the full quantitative results of other metrics (e.g., AP50 and AP75), more details are available in the appendix.

#### 4.3. Ablation study and visualization

**Weight initialization.** We explore two different ways of initializing the weights of the novel classifier before few-shot fine-tuning: (1) random initialization and (2) fine-tuning a

predictor on the novel set and using the classifier’s weights as initialization. We compare both methods on  $K = 1, 3, 10$  on split 3 of PASCAL VOC and COCO and show the results in Table 5. On PASCAL VOC, simple random initialization can outperform initialization using fine-tuned novel weights. On COCO, using the novel weights can improve the performance over random initialization. This is probably due to the increased complexity and number of classes of COCO compared to PASCAL VOC. We use random initialization for all PASCAL VOC experiments and novel initialization for all COCO and LVIS experiments.

**Scaling factor of cosine similarity.** We explore the effect of different scaling factors for computing cosine similarity. We compare three different factors,  $\alpha = 10, 20, 50$ . We use the same evaluation setting as the previous ablation study

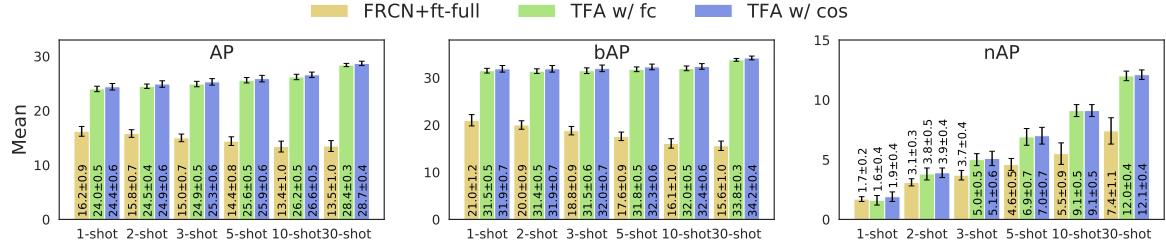


Figure 5. Generalized object detection benchmarks on COCO. For each metric, we report the average and 95% confidence interval computed over 10 random samples.

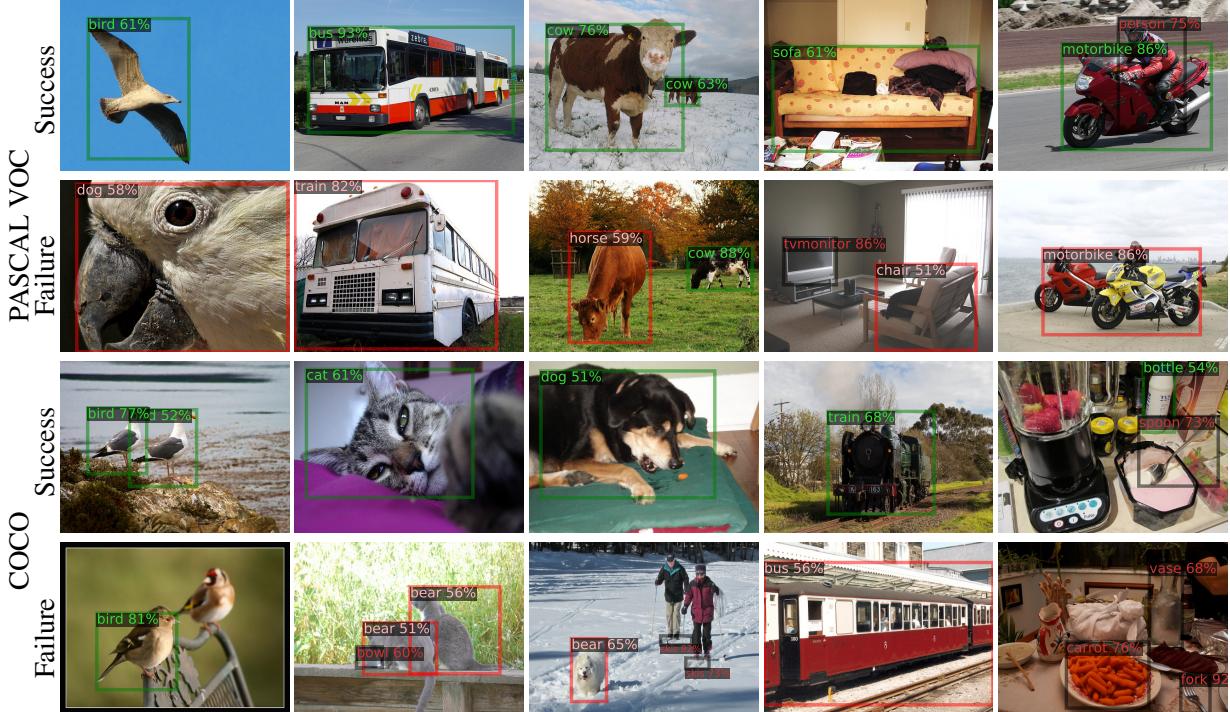


Figure 6. Success (green boxes) and failure (red boxes) cases of our approach on novel classes from split 1 of PASCAL VOC (bird, bus, cow, sofa, and motorbike) and COCO (bird, cat, dog, train, and bottle). The black boxes are detected objects of irrelevant classes, which can be ignored.

and report the results in Table 6. On PASCAL VOC,  $\alpha = 20$  outperforms the other scale factors in both base AP and novel AP. On COCO,  $\alpha = 20$  achieves better novel AP at the cost of worse base AP. Since it has the best performance on novel classes across both datasets, we use  $\alpha = 20$  in all of our experiments with cosine similarity.

**Detection results.** We provide qualitative visualizations of the detected novel objects on PASCAL VOC and COCO in Figure 6. We show both success (green boxes) and failure cases (red boxes) when detecting novel objects for each dataset to help analyze the possible error types. On the first split of PASCAL VOC, we visualize the results of our 10-shot TFA w/ cos model. On COCO, we visualize the

Table 5. Ablation of weight initialization of the novel classifier.

| Dataset                 | Init.  | Base AP     |             |             | Novel AP    |             |             |
|-------------------------|--------|-------------|-------------|-------------|-------------|-------------|-------------|
|                         |        | 1           | 3           | 10          | 1           | 3           | 10          |
| PASCAL VOC<br>(split 3) | Random | <b>51.2</b> | 52.6        | <b>52.8</b> | <b>15.6</b> | <b>25.0</b> | <b>29.4</b> |
|                         | Novel  | 50.9        | <b>53.1</b> | 52.5        | 13.4        | 24.9        | 28.9        |
| COCO                    | Random | 34.0        | <b>34.7</b> | <b>34.6</b> | 3.2         | 6.4         | 9.6         |
|                         | Novel  | <b>34.1</b> | <b>34.7</b> | <b>34.6</b> | 3.4         | <b>6.6</b>  | <b>9.8</b>  |

results of the 30-shot TFA w/ cos model. The failure cases include misclassifying novel objects as similar base objects, e.g., row 2 columns 1, 2, 3, and 4, mislocalizing the objects, e.g., row 2 column 5, and missing detections, e.g., row 4 columns 1 and 5.

**Table 6.** Ablation of scaling factor of cosine similarity.

| Dataset                 | Scale | Base AP     |             |             | Novel AP    |             |             |
|-------------------------|-------|-------------|-------------|-------------|-------------|-------------|-------------|
|                         |       | 1           | 3           | 10          | 1           | 3           | 10          |
| PASCAL VOC<br>(split 3) | 10    | 51.0        | <b>52.8</b> | 52.7        | 9.9         | 24.9        | 19.4        |
|                         | 20    | <b>51.2</b> | 52.6        | <b>52.8</b> | <b>15.6</b> | <b>25.0</b> | <b>29.4</b> |
|                         | 50    | 47.4        | 48.7        | 50.4        | 13.2        | 22.5        | 27.6        |
| COCO                    | 10    | <b>34.3</b> | <b>34.9</b> | <b>35.0</b> | 2.8         | 3.4         | 4.7         |
|                         | 20    | 34.1        | 34.7        | 33.9        | <b>3.4</b>  | <b>6.6</b>  | <b>10.0</b> |
|                         | 50    | 30.1        | 30.7        | 34.3        | 2.4         | 5.4         | 9.0         |

## 5. Conclusion

We proposed a simple two-stage fine-tuning approach for few-shot object detection. Our method outperformed the previous meta-learning methods by a large margin on the current benchmarks. In addition, we built more reliable benchmarks with revised evaluation protocols. On the new benchmarks, our models achieved new states of the arts, and on the LVIS dataset our models improved the AP of rare classes by 4 points with negligible reduction of the AP of frequent classes.

### ACKNOWLEDGMENTS

This work was supported by Berkeley AI Research, RISE Lab, Berkeley DeepDrive and DARPA.

## References

- Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.
- Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*, 2019.
- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>, 2007.
- Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.
- Gidaris, S. and Komodakis, N. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4367–4375, 2018.
- Gupta, A., Dollar, P., and Girshick, R. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5356–5364, 2019.
- Hariharan, B. and Girshick, R. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3018–3027, 2017.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., and Darrell, T. Few-shot object detection via feature reweighting. In *ICCV*, 2019.
- Koch, G. Siamese neural networks for one-shot image recognition. In *Proceedings of the 32 nd International Conference on Machine Learning*, 2015.
- Landau, B., Smith, L. B., and Jones, S. S. The importance of shape in early lexical learning. *Cognitive development*, 3(3):299–321, 1988.
- Lin, T.-Y., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *ECCV*, 2014.
- Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.
- Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.
- Qi, H., Brown, M., and Lowe, D. G. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5822–5830, 2018.
- Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.
- Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.
- Samuelson, L. K. and Smith, L. B. They call it like they see it: Spontaneous naming and attention to shape. *Developmental Science*, 8(2):182–198, 2005.
- Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Smith, L. B., Jones, S. S., Landau, B., Gershkoff-Stowe, L., and Samuelson, L. Object name learning provides on-the-job training for attention. *Psychological science*, 13(1):13–19, 2002.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087, 2017.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Wang, X., Yu, F., Wang, R., Darrell, T., and Gonzalez, J. E. Tafe-net: Task-aware feature embeddings for low shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1831–1840, 2019a.

Wang, Y.-X., Ramanan, D., and Hebert, M. Meta-learning to detect rare objects. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9925–9934, 2019b.

Yan, X., Chen, Z., Xu, A., Wang, X., Liang, X., and Lin, L. Meta r-cnn: Towards general solver for instance-level low-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9577–9586, 2019.

## A. Generalized Object Detection Benchmarks

We present the full benchmark results of PASCAL VOC (Table 7) and COCO (Table 8) on the revised benchmark used in this work. We report the average AP, AP50 and AP75 for all the classes, base classes only, and novel classes only in the tables. For each evaluation metric, we report the average value of  $n$  repeated runs with different groups of randomly sampled training shots (30 for PASCAL VOC and 10 for COCO) as well as the 95% confidence interval estimate of the mean values. The 95% confidence interval is calculated by

$$95\% CI = 1.96 \cdot \frac{s}{\sqrt{n}}, \quad (3)$$

where 1.96 is the  $Z$ -value,  $s$  is the standard deviation, and  $n$  is the number of repeated runs.

We compare two of our methods, one using a FC-based classifier (TFA w/ fc) and one using a cosine similarity based classifier (TFA w/ cos). We also compare against a fine-tuning baseline FRCN+ft-full and against FSRW (Kang et al., 2019) using their released code on PASCAL VOC shown in Table 7.

As shown in Table 7, TFA w/ cos is able to significantly outperform TFA w/ fc in overall AP across most splits

and shots. We observe that using a cosine similarity based classifier can achieve much higher accuracy on base classes, especially in higher shots. On split 1 and 3, TFA w/ cos is able to outperform TFA w/ fc by over 3 points on bAP on 5 and 10 shots. Across all shots in split 1, TFA w/ cos consistently outperforms TFA w/ fc on nAP75 by over 2 points in the novel classes.

Moreover, the AP of our models is usually over 10 points higher than that of FRCN+ft-full and FSRW on all settings. Note that FSRW uses YOLOv2 as the base object detector, while we are using Faster R-CNN. Wang et al. (2019b) shows that there are only about 2 points of difference when using a one or two-stage detector. Therefore, our improvements should still be significant despite the difference in the base detector.

We evaluate on COCO over six different number of shots  $K = 1, 2, 3, 5, 10, 30$  shown in Table 8. Although the differences are less significant than on PASCAL VOC, similar observations can be made about accuracy on base classes and novel classes.

## B. Performance over Multiple Runs

In our revised benchmark, we adopt  $n$  repeated runs with different randomly sampled training shots to increase the reliability of the benchmark. In our experiments, we adopt  $n = 30$  for PASCAL VOC and  $n = 10$  for COCO.

In this section, we provide plots of cumulative means with 95% confidence intervals of the repeated runs to show that the selected value of  $n$  is sufficient to provide statistically stable results.

We plot the model performance measured by AP, AP50 and AP75 of up to 40 random groups of training shots across all three splits in Figure 7. For COCO, we plot up to 10 random groups of training shots in Figure 8.

As we can observe from both Figure 7 and Figure 8, after around 30 runs on PASCAL VOC and 8 runs on COCO, the means and variances stabilize and our selected values of  $n$  are sufficient to obtain stable estimates of the model performances and reliable comparisons across different methods.

We also observe that the average value across multiple runs is consistently lower than that on the first run, especially in the one-shot case. For example, the average AP50 across 40 runs is around 15 points lower than the AP50 on the first run in the 1-shot case on split 1 on PASCAL VOC. This indicates that the accuracies on the first run, adopted by the previous work (Kang et al., 2019; Yan et al., 2019; Wang et al., 2019b), often overestimate the actual performance and thus lead to unreliable comparison between different approaches.



Table 8. Generalized object detection benchmarks on COCO. For each metric, we report the average and 95% confidence interval computed over 10 random samples.

| # shots | Method       | Overall  |          |          |          |          |          | Base class |          |          | Novel class |          |          |
|---------|--------------|----------|----------|----------|----------|----------|----------|------------|----------|----------|-------------|----------|----------|
|         |              | AP       | AP50     | AP75     | APs      | APm      | API      | bAP        | bAP50    | bAP75    | nAP         | nAP50    | nAP75    |
| 1       | FRCN+ft-full | 16.2±0.9 | 25.8±1.2 | 17.6±1.0 | 7.2±0.6  | 17.9±1.0 | 23.1±1.1 | 21.0±1.2   | 33.3±1.7 | 23.0±1.4 | 1.7±0.2     | 3.3±0.3  | 1.6±0.2  |
|         | TFA w/fc     | 24.0±0.5 | 38.9±0.5 | 25.8±0.6 | 13.8±0.4 | 26.6±0.4 | 32.0±0.6 | 31.5±0.5   | 50.7±0.6 | 33.9±0.8 | 1.6±0.4     | 3.4±0.6  | 1.3±0.4  |
|         | TFA w/cos    | 24.4±0.6 | 39.8±0.8 | 26.1±0.8 | 14.7±0.7 | 26.8±0.5 | 31.4±0.7 | 31.9±0.7   | 51.8±0.9 | 34.3±0.9 | 1.9±0.4     | 3.8±0.6  | 1.7±0.5  |
| 2       | FRCN+ft-full | 15.8±0.7 | 25.0±1.1 | 17.3±0.7 | 6.6±0.6  | 17.2±0.8 | 23.5±0.7 | 20.0±0.9   | 31.4±1.5 | 22.2±1.0 | 3.1±0.3     | 6.1±0.6  | 2.9±0.3  |
|         | TFA w/fc     | 24.5±0.4 | 39.3±0.6 | 26.5±0.5 | 13.9±0.3 | 27.1±0.5 | 32.7±0.7 | 31.4±0.5   | 49.8±0.7 | 34.3±0.6 | 3.8±0.5     | 7.8±0.8  | 3.2±0.6  |
|         | TFA w/cos    | 24.9±0.6 | 40.1±0.9 | 27.0±0.7 | 14.9±0.7 | 27.3±0.6 | 32.3±0.6 | 31.9±0.7   | 50.8±1.1 | 34.8±0.8 | 3.9±0.4     | 7.8±0.7  | 3.6±0.6  |
| 3       | FRCN+ft-full | 15.0±0.7 | 23.9±1.2 | 16.4±0.7 | 6.0±0.6  | 16.1±0.9 | 22.6±0.9 | 18.8±0.9   | 29.5±1.5 | 20.7±0.9 | 3.7±0.4     | 7.1±0.8  | 3.5±0.4  |
|         | TFA w/fc     | 24.9±0.5 | 39.7±0.7 | 27.1±0.6 | 14.1±0.4 | 27.5±0.6 | 33.4±0.8 | 31.5±0.6   | 49.6±0.7 | 34.6±0.7 | 5.0±0.5     | 9.9±1.0  | 4.6±0.6  |
|         | TFA w/cos    | 25.3±0.6 | 40.4±1.0 | 27.6±0.7 | 14.8±0.7 | 27.7±0.6 | 33.1±0.7 | 32.0±0.7   | 50.5±1.0 | 35.1±0.7 | 5.1±0.6     | 9.9±0.9  | 4.8±0.6  |
| 5       | FRCN+ft-full | 14.4±0.8 | 23.0±1.3 | 15.6±0.8 | 5.6±0.4  | 15.2±1.0 | 21.9±1.1 | 17.6±0.9   | 27.8±1.5 | 19.3±1.0 | 4.6±0.5     | 8.7±1.0  | 4.4±0.6  |
|         | TFA w/fc     | 25.6±0.5 | 40.7±0.8 | 28.0±0.5 | 14.3±0.4 | 28.2±0.6 | 34.4±0.6 | 31.8±0.5   | 49.8±0.7 | 35.2±0.5 | 6.9±0.7     | 13.4±1.2 | 6.3±0.8  |
|         | TFA w/cos    | 25.9±0.6 | 41.2±0.9 | 28.4±0.6 | 15.0±0.6 | 28.3±0.5 | 34.1±0.6 | 32.3±0.6   | 50.5±0.9 | 35.6±0.6 | 7.0±0.7     | 13.3±1.2 | 6.5±0.7  |
| 10      | FRCN+ft-full | 13.4±1.0 | 21.8±1.7 | 14.5±0.9 | 5.1±0.4  | 14.3±1.2 | 20.1±1.5 | 16.1±1.0   | 25.7±1.8 | 17.5±1.0 | 5.5±0.9     | 10.0±1.6 | 5.5±0.9  |
|         | TFA w/fc     | 26.2±0.5 | 41.8±0.7 | 28.6±0.5 | 14.5±0.3 | 29.0±0.5 | 35.2±0.6 | 32.0±0.5   | 49.9±0.7 | 35.3±0.6 | 9.1±0.5     | 17.3±1.0 | 8.5±0.5  |
|         | TFA w/cos    | 26.6±0.5 | 42.2±0.8 | 29.0±0.6 | 15.0±0.5 | 29.1±0.4 | 35.2±0.5 | 32.4±0.6   | 50.6±0.9 | 35.7±0.7 | 9.1±0.5     | 17.1±1.1 | 8.8±0.5  |
| 30      | FRCN+ft-full | 13.5±1.0 | 21.8±1.9 | 14.5±1.0 | 5.1±0.3  | 14.6±1.2 | 19.9±2.0 | 15.6±1.0   | 24.8±1.8 | 16.9±1.0 | 7.4±1.1     | 13.1±2.1 | 7.4±1.0  |
|         | TFA w/fc     | 28.4±0.3 | 44.4±0.6 | 31.2±0.3 | 15.7±0.3 | 31.2±0.3 | 38.6±0.4 | 33.8±0.3   | 51.8±0.6 | 37.6±0.4 | 12.0±0.4    | 22.2±0.6 | 11.8±0.4 |
|         | TFA w/cos    | 28.7±0.4 | 44.7±0.7 | 31.5±0.4 | 16.1±0.4 | 31.2±0.3 | 38.4±0.4 | 34.2±0.4   | 52.3±0.7 | 38.0±0.4 | 12.1±0.4    | 22.0±0.7 | 12.0±0.5 |

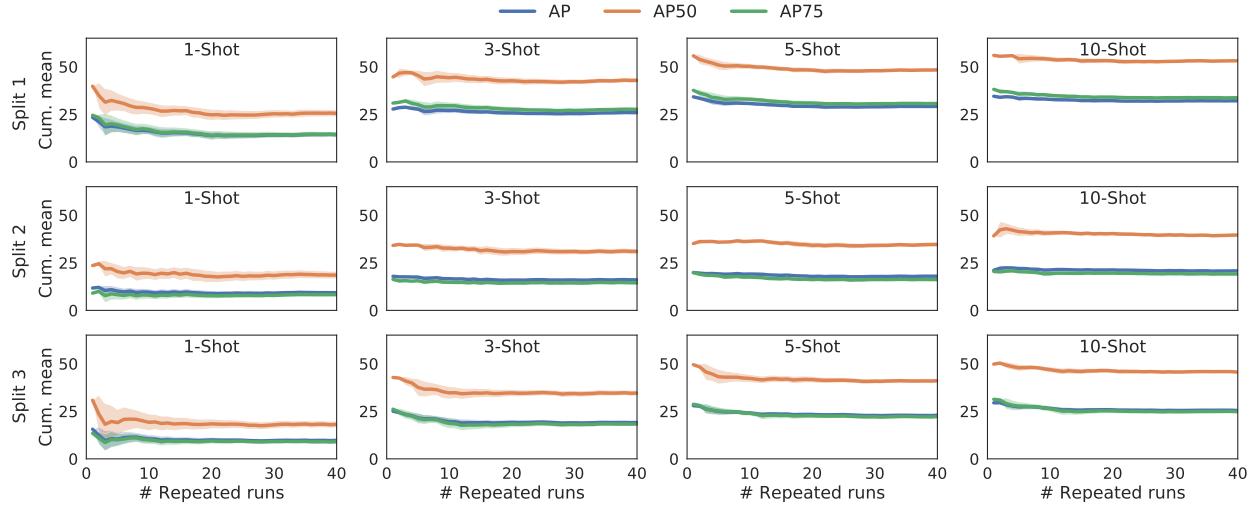


Figure 7. Cumulative means with 95% confidence intervals across 40 repeated runs, computed on the novel classes of all three splits of PASCAL VOC. The means and variances become stable after around 30 runs.

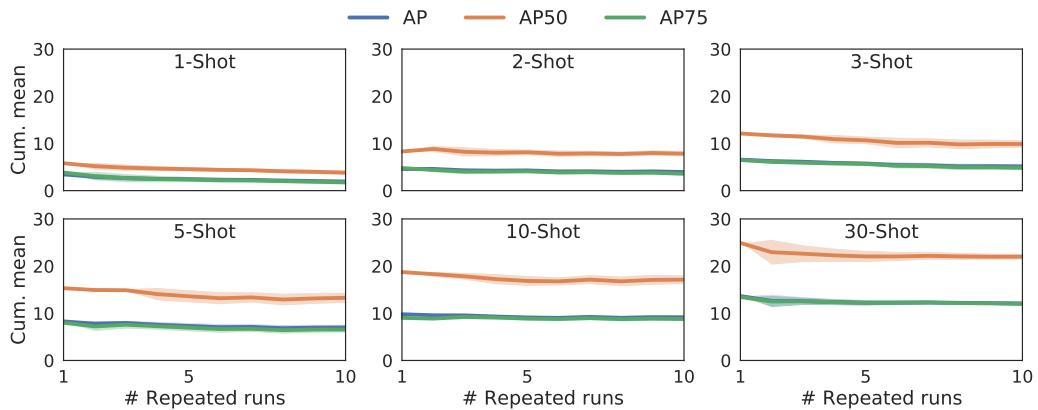


Figure 8. Cumulative means with 95% confidence intervals across 10 repeated runs, computed on the novel classes of COCO.