CS282 Machine Learning Report

Junda Shen [*1] **Zian He** [*1]

Abstract

In this project, we try to conquer the few-shot object detection problem by fine-tuning the last layer of some existing convolutional object detectors, e.g. Faster R-CNN. The prior methods usually encounter high variance problems which made them unreliable. However, we used an elegant sampling method so that the results are stable and the overfitting problem is diminished. We evaluated our approach on COCO and LVIS datasets, which shows that our method outperforms prior methods by $2 \sim 15$ percent. Since this method does not propose a new architecture, we implement it based on the Detectron2 package, which already contains many pretrained CNN architectures and is best suited for a fine-tuning task.

1. Introduction

Nowadays, many tasks can be done by machine perception systems and learning from only a few samples has raised wide attention due to the scarcity of samples in some particular areas. Although some few-shot learning problems has almost been solved, it is still hard to achieve image detection with few training samples, the detection results are still unideal.

In the prior works, people have tried to achieve few-shot image detection by two approaches

1. Meta-learning. This approach tries to transfer knowledge learnt from data-abundent classes to data-scarce novel classes so that the model could adapt to the new task more quickly. This method have been used in few-shot image classification for a while but it has not been proved effective in object detection task, which is much more challenging than image classification.

2. Metric-learning. This approach tries to build some metrics for the learner so that it can estimate and check the similarities between images, e.g. cosine similarities. Based on the built metrics, the model can try to fit to the input and learn how to detect objects. However, this approach relis on the metric, which may be hard to find a better one.

[*]Equal contribution [1]SIST, Shanghaitech University. Correspondence to: Junda Shen <shenjd@shanghaitech.edu.cn>, Zian He <heza@shanghaitech.edu.cn>.

The object detection task is much harder than the image classification task, since it contains not only classification but also object localization. Researchers (Kang et al., 2019; Yan et al., 2019; Wang et al., 2019b) have attempted to achieve few-shot object detection task, where a few labeled data, far from abundent, is provided to the learner as novel training data. These methods achieved out-of-random performance but their methods are not stable, which means the reproduced results might be far from their reported results.

In this project, we try to improve the detection results by adopting fine-tuning based approaches. We focus on the schedule of the training procedure and use proper instance-level feature normalization in this project.

The training is composed of two stages, which is shown in Figure 1. The first stage trains the whole network, e.g. Faster R-CNN, on data-abundent base classes. Then we only train the last layer of the network in the second stage, on a small balanced training set that is composed of both base and novel classes by properly sample data, parameters in other layers of the detector is kept intact during the second stage. We also use instance-level feature normalization in the second stage, which helps the model to diminish some parameter issues introduced by Gidaris & Komodakis (2018); Qi et al. (2018); Chen et al. (2019).

In our evalution, we find that our method outperfroms all previous few-shot object detection methods, our method can gain $2 \sim 15$ percent more accuracy than prior methods, and it even doubled the accuracy in one-shot learning, which dictates the effiction of our method.

We fixed several issues in the existing training procedures

1. We try to diminish overfitting by invoking multiple runs on different random samples so that the variance is lower and the results much are more stable.

2. We try to keep the accuracy of our method consistent with the paper.

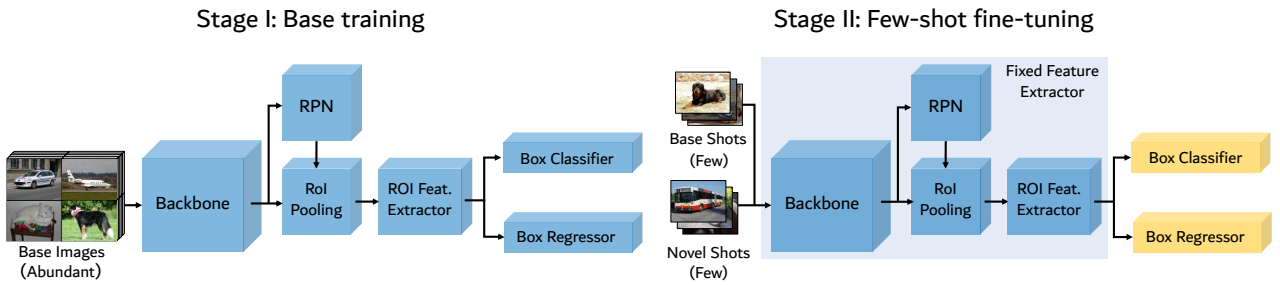

Figure 1. The two-stage fine-tuning approach (TFA). In the base training stage, the entire object detector are jointly trained on the base classes. In the second stage, the labeled feature extractor is kept intact and only the last layer, i.e. box predictor, is fine-tuned on a balanced set consisting sampled base and novel classes.

3. We report not only the accuracy for novel classes, but also accuracy for base classes and the overall accuracy for all classes, so that one can see that the accuracy (referred to as the generalized few-shot learning setting in the few-shot classification literature (Hariharan & Girshick, 2017; Wang et al., 2019a)) for other classes are not largely affected.

2. Algorithms for Few-Shot Object Detection

In this section, we first focus on the few-shot object detection setting. Then, we will show our two-stage fine-tuning approach in Section 2.1 and Section 2.2.

According to Kang et al. (2019), We first adjust the few-shot object detection settings. We use 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 $\mathcal{D} = \{(x, y), x \in \mathcal{X}, y \in \mathcal{Y}\}$ (x is the input image, $y = \{(c_i, \mathbf{l}_i), i = 1, ..., N\}$ denotes the categories $c \in C_b \cup C_n$ and bounding box coordinates $\mathbf{l}$ of the N object instances in the image x).

For COCO, we balcance the novel set and each class from it has the same number of annotated objects (*i.e.*, K-shot).

For LVIS, it has a natural long-tail distribution, which does not have the manual K-shot split. To solve this problem, we divide the classes in LVIS into three classes, *frequent* classes (appearing in more than 100 images), *common* classes (10-100 images), and *rare* classes (less than 10 images).

We evaluate our few-shot object detector on a test set with the base classes as well as 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.

2.1. Two-stage fine-tuning approach

Our fine-tuning approach (TFA) including two stages. The base detection model is formed by Faster R-CNN (Ren et al., 2015) and a two-stage object detector.

As shown in Figure 1, the feature learning components ($\mathcal{F}$) 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. This stage we use a large number of basic data samples to train ordinary 2D target detection networks (*e.g.*, Faster-RCNN). This stage is the traditional training method. The loss of the network consists of three parts,

$$\mathcal{L} = \mathcal{L}_{\text{rpn}} + \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{loc}}, \tag{1}$$

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

Few-shot fine-tuning. The second stage is fine-tuning based on a small sample. While keeping the entire feature extractor unchanged, assign the randomly initialized weights of the new class to the box prediction network, and only fine-tune the box classification and regression network, which is the last layer of the detection model. This process uses the same loss function (in Equation 1) as the first stage and reduces the learning rate. The learning rate is reduced by 20 from the first stage in all our experiments.

Cosine similarity for box classifier. The design of the classifier is based on the cosine similarity function, and the formula is shown in in Equation 2, inspired by Gidaris & Komodakis (2018); Qi et al. (2018); Chen et al. (2019). The

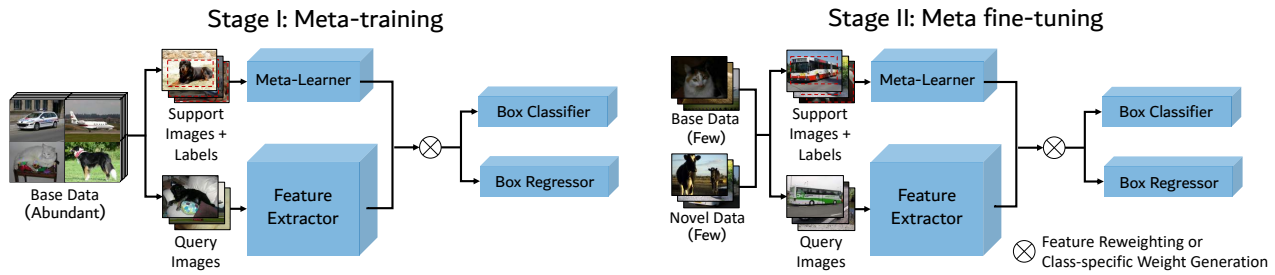

Figure 2. Abstraction of the meta-learning based few-shot object detectors. A meta-learner is introduced to acquire task-level meta information and help the model generalize to novel classes through feature re-weighting (*e.g.*, FSRW and Meta R-CNN) or weight generation (*e.g.*, MetaDet). A two-stage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

weight matrix $W \in \mathbb{R}^{d\times c}$ of the box classifier $\mathcal{C}$ can be written as $[w_1, w_2, ..., w_c]$, where $w_c \in \mathbb{R}^d$ is the per-class weight vector. The output of $\mathcal{C}$ is scaled similarity scores S of the input feature $\mathcal{F}(x)$ and the weight vectors of different classes. The entries in S are

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

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. α is the scaling factor. We use a fixed α of 20 in our experiments. Compared with the FC-based classifier, the cosine similarity classifier based on instance-level feature normalization can help reduce the variance of the novel class, improve detection accuracy, especially in When the number of training samples is small.

2.2. Compared Meta-learning based approaches with Fine-tuning approach

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 when the supporting set increases. Since 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.

3. Experiments

We evaluate our method on two datasets

1. COCO
2. LVIS

We then provide some quantitative results and visualization results in this section.

Implementation details. We select Faster R-CNN (Ren et al., 2015) as our base detector and use Resnet-101 (He et al., 2016) with a Feature Pyramid Network (Lin et al., 2017) as the backbone of our method. 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 hyperparameters related to the model architecture, we use the default parameters provided by Detectron2.

3.1. 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 evaluate our approach on COCO, using the same data splits and training examples provided by Kang et al. (2019). For COCO, the 60 categoriesare used as base classes while the remaining 20 classes are used as novel classes with $K = 10, 30$. For evaluation metrics, 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 COCO. Similarly, we report the average AP and AP75 of the 20 novel classes on COCO in Table 1. 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.

Table 1. 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)	**10.0**	13.4	9.2	13.2
TFA w/ cos (Ours)	**10.0**	**13.7**	**9.3**	**13.4**

3.2. 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.

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 2. Compared to the methods in Gupta et al. (2019), our approach is able to achieve better performance of ∼1-1.5 points in overall AP and ∼2-4 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 ∼2-3 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 COCO. We show evaluation results on COCO in Figure 3. we evaluate on the base classes and the novel classes and report AP scores. 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.

3.3. 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 COCO and show the results in Table 3. 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. We use 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 and report the results in Table 4. 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 COCO in Figure 4. We show both success (green boxes) and failure cases (red boxes) when detecting novel objects for COCO to help analyze the possible error types. On COCO, we visualize the 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.

4. 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

Table 2. 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	APl	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	**28.5**
TFA w/ fc (Ours)			22.6	37.5	22.6	17.5	27.2	36.3	14.2	21.0	26.8
TFA w/ cos (Ours)			22.9	37.5	23.6	19.0	27.5	37.0	15.5	20.8	27.9
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)			24.0	40.0	26.0	19.3	28.9	36.5	15.0	24.1	28.1
TFA w/ cos (Ours)			**24.5**	**40.2**	**26.4**	**20.1**	**29.5**	**38.5**	**17.1**	**24.5**	27.8
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	**30.8**
TFA w/ fc (Ours)			24.0	39.2	25.1	19.2	29.8	38.8	15.8	22.7	29.1
TFA w/ cos (Ours)			24.5	39.6	26.0	20.3	30.6	39.8	**18.3**	21.5	30.1
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)			25.6	**41.8**	26.8	20.0	31.0	39.3	15.7	26.1	28.3
TFA w/ cos (Ours)			**26.5**	**41.9**	**27.6**	**20.1**	**32.2**	**40.0**	17.2	**26.3**	29.8

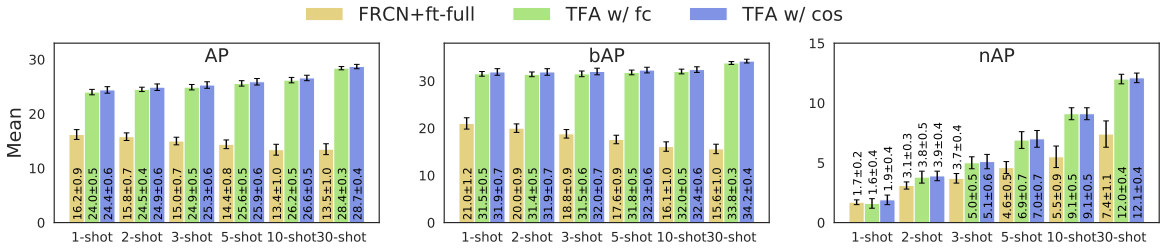

Figure 3. Generalized object detection benchmarks on COCO. For each metric, we report the average and 95% confidence interval computed over 10 random samples.

Table 3. Ablation of weight initialization of the novel classifier.

Dataset	Init.	Base AP 1	Base AP 3	Base AP 10	Novel AP 1	Novel AP 3	Novel AP 10
COCO	Random	34.0	**34.7**	**34.6**	3.2	6.4	9.6
	Novel	**34.1**	**34.7**	**34.6**	**3.4**	**6.6**	**9.8**

Table 4. Ablation of scaling factor of cosine similarity.

Dataset	Scale	Base AP 1	Base AP 3	Base AP 10	Novel AP 1	Novel AP 3	Novel AP 10
COCO	10	**34.3**	**34.9**	**35.0**	2.8	3.4	4.7
	20	34.1	34.7	33.9	**3.4**	**6.6**	**10.0**
	50	30.1	30.7	34.3	2.4	5.4	9.0

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.

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.

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*

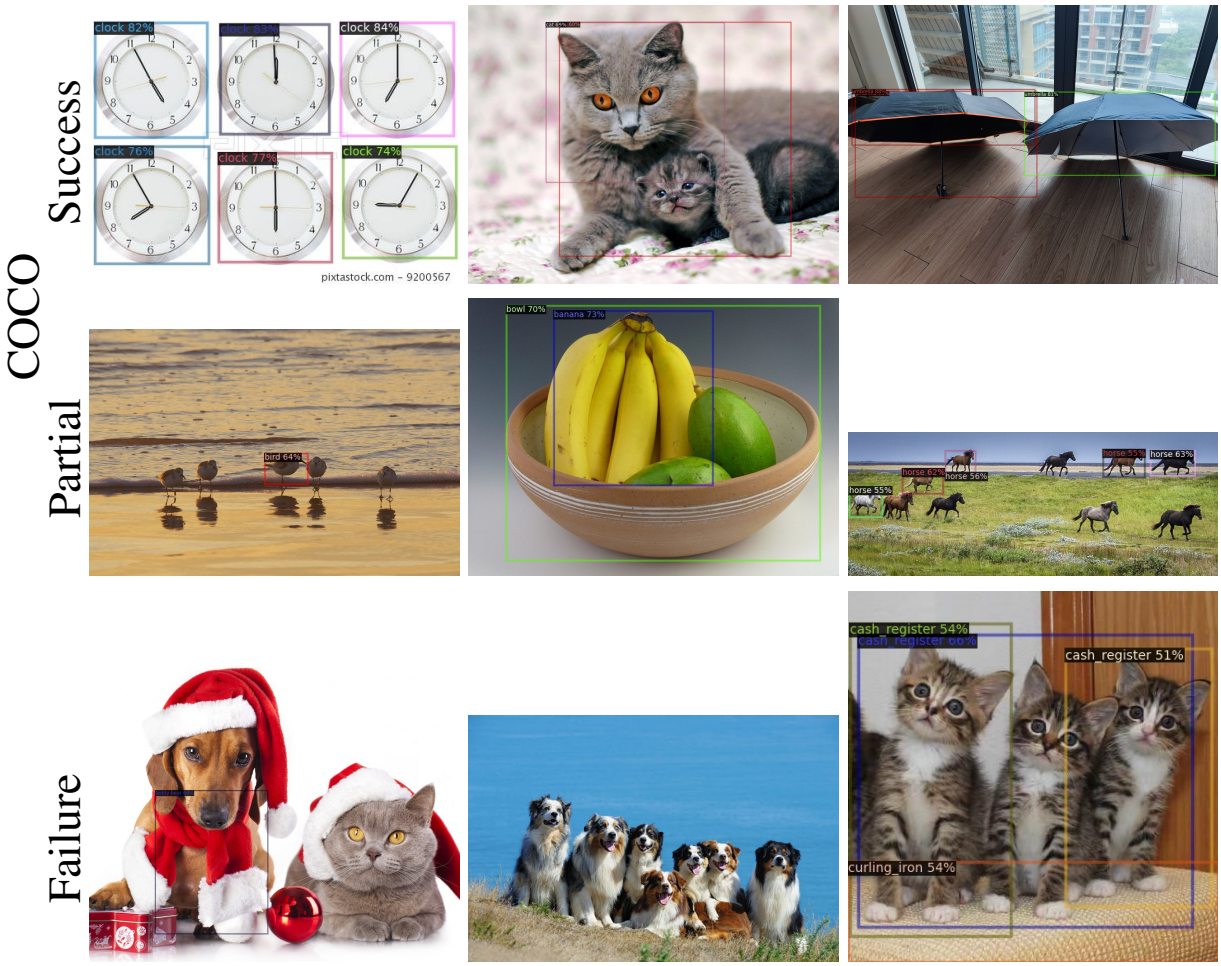

Figure 4. Success partial and failure cases of our approach on novel classes from COCO. The black boxes are detected objects of irrelevant classes, which can be ignored.

computer vision and pattern recognition, pp. 2117–2125, 2017.

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.

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

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 5) and COCO (Table 6) 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

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

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 5.

As shown in Table 5, `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 6. 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 5. For COCO, we plot up to 10 random groups of training shots in Figure 6.

As we can observe from both Figure 5 and Figure 6, 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 5. Generalized object detection benchmarks on PASCAL VOC. For each metric, we report the average and 95% confidence interval computed over 30 random samples.

Split	# shots	Method	Overall			Base class			Novel class		
			AP	AP50	AP75	bAP	bAP50	bAP75	nAP	nAP50	nAP75
Split 1	1	FSRW (Kang et al., 2019)	27.6 ± 0.5	50.8 ± 0.9	26.5 ± 0.6	34.1 ± 0.5	62.9 ± 0.9	32.6 ± 0.5	8.0 ± 1.0	14.2 ± 1.7	7.9 ± 1.1
		FRCN+ft-full	30.2±0.6	49.4±0.7	32.2±0.9	38.2±0.8	62.6±1.0	40.8±1.1	6.0±0.7	9.9±1.2	6.3±0.8
		TFA w/fc	39.6±0.5	63.5±0.7	43.2±0.7	48.7±0.7	77.1±0.7	53.7±1.0	12.2±1.6	22.9±2.5	11.6±1.9
		TFA w/cos	40.6±0.5	64.5±0.6	44.7±0.6	49.4±0.4	77.6±0.2	54.8±0.5	14.2±1.4	25.3±2.2	14.2±1.8
	2	FSRW (Kang et al., 2019)	28.7±0.4	52.2±0.6	27.7±0.5	33.9±0.4	61.8±0.5	32.7±0.5	13.2±1.0	23.6±1.7	12.7±1.1
		FRCN+ft-full	30.5±0.6	49.4±0.8	32.6±0.7	37.3±0.7	60.7±1.0	40.1±0.9	9.9±0.9	15.6±1.4	10.3±1.0
		TFA w/fc	40.5±0.5	65.5±0.7	43.8±0.7	47.8±0.7	75.8±0.7	52.2±1.0	18.9±1.5	34.5±2.4	18.4±1.9
		TFA w/cos	42.6±0.3	67.1±0.4	47.0±0.4	49.6±0.3	77.3±0.2	55.0±0.4	21.7±1.0	36.4±1.6	22.8±1.3
	3	FSRW (Kang et al., 2019)	29.5±0.3	53.3±0.6	28.6±0.4	33.8±0.3	61.2±0.6	32.7±0.4	16.8±0.9	29.8±1.6	16.5±1.0
		FRCN+ft-full	31.8±0.5	51.4±0.8	34.2±0.6	37.9±0.5	61.3±0.7	40.7±0.6	13.7±1.0	21.6±1.6	14.8±1.1
		TFA w/fc	41.8±0.9	67.1±0.9	45.4±1.2	48.2±0.9	76.0±0.9	53.1±1.2	22.6±1.2	40.4±1.7	22.4±1.7
		TFA w/cos	43.7±0.3	68.5±0.4	48.3±0.4	49.8±0.3	77.3±0.2	55.4±0.4	25.4±0.9	42.1±1.5	27.0±1.2
	5	FSRW (Kang et al., 2019)	30.4±0.3	54.6±0.5	29.6±0.4	33.7±0.3	60.7±0.4	32.8±0.4	20.6±0.8	36.5±1.4	20.0±0.9
		FRCN+ft-full	32.7±0.5	52.5±0.8	35.0±0.6	37.6±0.4	60.6±0.6	40.3±0.5	17.9±1.1	28.0±1.7	19.2±1.3
		TFA w/fc	41.9±0.6	68.0±0.7	45.0±0.8	47.2±0.6	75.1±0.6	51.5±0.8	25.9±1.0	46.7±1.4	25.3±1.2
		TFA w/cos	44.8±0.3	70.1±0.4	49.4±0.4	50.1±0.2	77.4±0.3	55.6±0.3	28.9±0.8	47.9±1.2	30.6±1.0
	10	FRCN+ft-full	33.3±0.4	53.8±0.6	35.5±0.4	36.8±0.4	59.8±0.6	39.2±0.4	22.7±0.9	35.6±1.5	24.4±1.0
		TFA w/fc	42.8±0.3	69.5±0.4	46.0±0.4	47.3±0.3	75.4±0.3	51.6±0.4	29.3±0.7	52.0±1.1	29.0±0.9
		TFA w/cos	45.8±0.2	71.3±0.3	50.4±0.3	50.4±0.2	77.5±0.2	55.9±0.3	32.0±0.6	52.8±1.0	33.7±0.7
Split 2	1	FSRW (Kang et al., 2019)	28.4±0.5	51.7±0.9	27.3±0.6	35.7±0.5	64.8±0.9	34.6±0.7	6.3±0.9	12.3±1.9	5.5±0.7
		FRCN+ft-full	30.3±0.5	49.7±0.5	32.3±0.7	38.8±0.6	63.2±0.7	41.6±0.9	5.0±0.6	9.4±1.2	4.5±0.7
		TFA w/fc	36.2±0.8	59.6±0.9	38.7±1.0	45.6±0.9	73.8±0.9	49.4±1.2	8.1±1.2	16.9±2.3	6.6±1.1
		TFA w/cos	36.7±0.6	59.9±0.8	39.3±0.8	45.9±0.7	73.8±0.8	49.8±1.1	9.0±1.2	18.3±2.4	7.8±1.2
	2	FSRW (Kang et al., 2019)	29.4±0.3	53.1±0.6	28.5±0.4	35.8±0.4	64.2±0.6	35.1±0.5	9.9±0.7	19.6±1.3	8.8±0.6
		FRCN+ft-full	30.7±0.5	49.7±0.7	32.9±0.6	38.4±0.5	61.6±0.7	41.4±0.7	7.7±0.8	13.8±1.4	7.4±0.8
		TFA w/fc	38.5±0.5	62.8±0.6	41.2±0.6	46.9±0.5	74.9±0.5	51.2±0.7	13.1±1.0	26.4±1.9	11.3±1.1
		TFA w/cos	39.0±0.4	63.0±0.5	42.1±0.6	47.3±0.4	74.9±0.4	51.9±0.7	14.1±0.9	27.5±1.6	12.7±1.0
	3	FSRW (Kang et al., 2019)	29.9±0.3	53.9±0.4	29.0±0.4	35.7±0.3	63.5±0.4	35.1±0.4	12.5±0.7	25.1±1.4	10.4±0.7
		FRCN+ft-full	31.1±0.3	50.1±0.5	33.2±0.5	38.1±0.4	61.0±0.6	41.2±0.5	9.8±0.9	17.4±1.6	9.4±1.0
		TFA w/fc	39.4±0.4	64.2±0.5	42.0±0.5	47.5±0.4	75.4±0.5	51.7±0.6	15.2±0.8	30.5±1.5	13.1±0.8
		TFA w/cos	40.1±0.3	64.5±0.5	43.3±0.4	48.1±0.3	75.6±0.4	52.9±0.5	16.0±0.8	30.9±1.6	14.4±0.9
	5	FSRW (Kang et al., 2019)	30.4±0.4	54.6±0.5	29.5±0.5	35.3±0.3	62.4±0.4	34.9±0.5	15.7±0.8	31.4±1.5	13.3±0.9
		FRCN+ft-full	31.5±0.3	50.8±0.7	33.6±0.4	37.9±0.4	60.4±0.6	40.8±0.5	12.4±0.9	21.9±1.5	12.1±0.9
		TFA w/fc	40.0±0.4	65.1±0.5	42.6±0.5	47.5±0.4	75.3±0.5	51.6±0.5	17.5±0.7	34.6±1.1	15.5±0.9
		TFA w/cos	40.9±0.4	65.7±0.5	44.1±0.5	48.6±0.4	76.2±0.4	53.3±0.5	17.8±0.8	34.1±1.4	16.2±1.0
	10	FRCN+ft-full	32.2±0.3	52.3±0.4	34.1±0.4	37.2±0.3	59.8±0.4	39.9±0.4	17.0±0.8	29.8±1.4	16.7±0.9
		TFA w/fc	41.3±0.2	67.0±0.3	44.0±0.3	48.3±0.2	76.1±0.3	52.7±0.4	20.2±0.5	39.7±0.9	18.0±0.7
		TFA w/cos	42.3±0.3	67.6±0.4	45.7±0.3	49.4±0.2	76.9±0.3	54.5±0.3	20.8±0.6	39.5±1.1	19.2±0.6
Split 3	1	FSRW (Kang et al., 2019)	27.5±0.6	50.0±1.0	26.8±0.7	34.5±0.7	62.5±1.2	33.5±0.7	6.7±1.0	12.5±1.6	6.4±1.0
		FRCN+ft-full	30.8±0.6	49.8±0.8	32.9±0.8	39.6±0.8	63.7±1.0	42.5±0.9	4.5±0.7	8.1±1.3	4.2±0.7
		TFA w/fc	39.0±0.6	62.3±0.7	42.1±0.8	49.5±0.8	77.8±0.8	54.0±1.0	7.8±1.1	15.7±2.1	6.5±1.0
		TFA w/cos	40.1±0.3	63.5±0.6	43.6±0.5	50.2±0.4	78.7±0.2	55.1±0.5	9.6±1.1	17.9±2.0	9.1±1.2
	2	FSRW (Kang et al., 2019)	28.7±0.4	51.8±0.7	28.1±0.5	34.5±0.4	62.0±0.7	34.0±0.5	11.3±0.7	21.3±1.0	10.6±0.8
		FRCN+ft-full	31.3±0.5	50.2±0.9	33.5±0.6	39.1±0.5	62.4±0.9	42.0±0.7	8.0±0.8	13.9±1.4	7.9±0.9
		TFA w/fc	41.1±0.6	65.1±0.7	44.3±0.7	50.1±0.7	77.7±0.7	54.8±0.9	14.2±1.2	27.2±2.0	12.6±1.3
		TFA w/cos	41.8±0.4	65.6±0.6	45.3±0.4	50.7±0.3	78.4±0.2	55.6±0.4	15.1±1.3	27.2±2.1	14.4±1.5
	3	FSRW (Kang et al., 2019)	29.2±0.4	52.7±0.6	28.5±0.4	34.2±0.3	61.3±0.6	33.6±0.4	14.2±0.7	26.8±1.4	13.1±0.7
		FRCN+ft-full	32.1±0.5	51.3±0.8	34.3±0.6	39.1±0.5	62.1±0.7	42.1±0.6	11.1±0.9	19.0±1.5	11.2±1.0
		TFA w/fc	40.4±0.5	65.4±0.7	43.1±0.7	47.8±0.5	75.6±0.5	52.1±0.7	18.1±1.0	34.7±1.6	16.2±1.3
		TFA w/cos	43.1±0.4	67.5±0.5	46.7±0.5	51.1±0.3	78.6±0.2	56.3±0.4	18.9±1.1	34.3±1.7	18.1±1.4
	5	FSRW (Kang et al., 2019)	30.1±0.3	53.8±0.5	29.3±0.4	34.1±0.3	60.5±0.4	33.6±0.4	18.0±0.7	33.8±1.4	16.5±0.8
		FRCN+ft-full	32.4±0.5	51.7±0.8	34.4±0.6	38.5±0.5	61.0±0.7	41.3±0.6	14.0±0.9	23.9±1.7	13.7±0.9
		TFA w/fc	41.3±0.5	67.1±0.6	44.0±0.6	48.0±0.5	75.8±0.5	52.2±0.6	21.4±0.9	40.8±1.3	19.4±1.0
		TFA w/cos	44.1±0.3	69.1±0.4	47.8±0.4	51.3±0.2	78.5±0.3	56.4±0.3	22.8±0.9	40.8±1.4	22.1±1.1
	10	FRCN+ft-full	33.1±0.5	53.1±0.7	35.2±0.5	38.0±0.5	60.5±0.7	40.7±0.6	18.4±0.8	31.0±1.2	18.7±1.0
		TFA w/fc	42.2±0.4	68.3±0.5	44.9±0.6	48.5±0.4	76.2±0.4	52.9±0.5	23.3±0.8	44.6±1.1	21.0±1.2
		TFA w/cos	45.0±0.3	70.3±0.4	48.9±0.4	51.6±0.2	78.6±0.2	57.0±0.3	25.4±0.7	45.6±1.1	24.7±1.1

Table 6. 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	APl	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

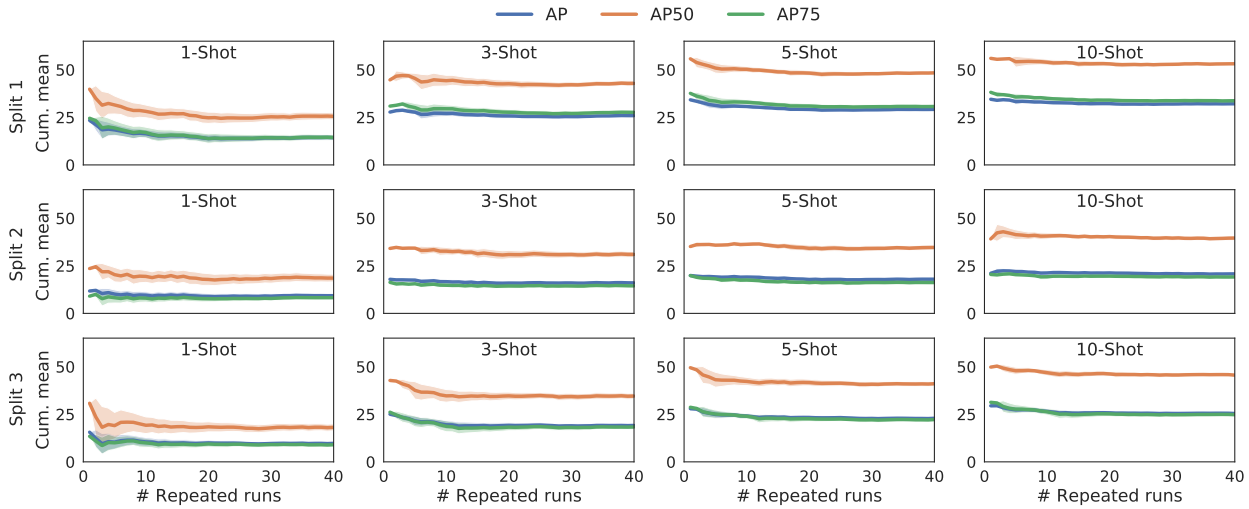

Figure 5. Cumulative means with 95% confidence intervals across 40 repeated runs, computed on the novel classes of all three splits of PASCAL VOC. The means and variances become stable after around 30 runs.

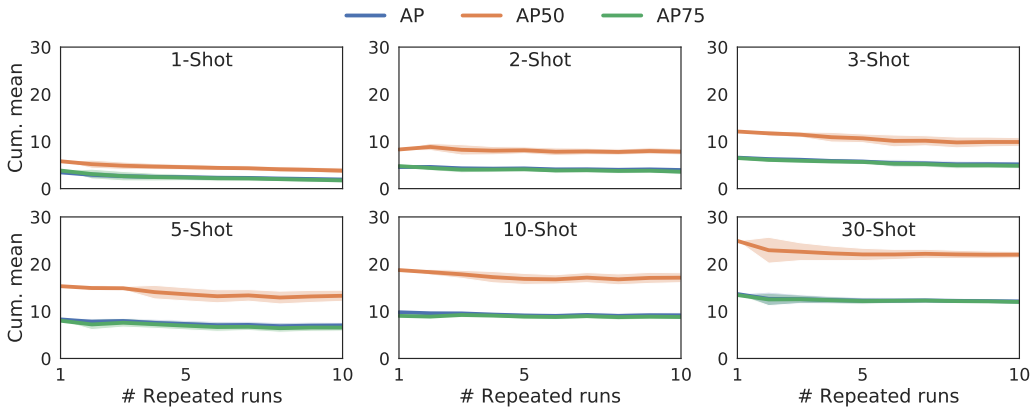

Figure 6. Cumulative means with 95% confidence intervals across 10 repeated runs, computed on the novel classes of COCO.