Rethinking Few-Shot Object Detection on a Multi-Domain Benchmark

Kibok Lee[1,2⋆], Hao Yang[1†], Satyaki Chakraborty[1], Zhaowei Cai[1], Gurumurthy Swaminathan[1], Avinash Ravichandran[1], and Onkar Dabeer[1]

[1]AWS AI Labs [2]Yonsei University
{kibok,haoyng,satyaki,zhaoweic,gurumurs,ravinash,onkardab}@amazon.com
kibok@yonsei.ac.kr

Abstract. Most existing works on few-shot object detection (FSOD) focus on a setting where both pre-training and few-shot learning datasets are from a similar domain. However, few-shot algorithms are important in multiple domains; hence evaluation needs to reflect the broad applications. We propose a Multi-dOmain Few-Shot Object Detection (MoFSOD) benchmark consisting of 10 datasets from a wide range of domains to evaluate FSOD algorithms. We comprehensively analyze the impacts of freezing layers, different architectures, and different pre-training datasets on FSOD performance. Our empirical results show several key factors that have not been explored in previous works: 1) contrary to previous belief, on a multi-domain benchmark, fine-tuning (FT) is a strong baseline for FSOD, performing on par or better than the state-of-the-art (SOTA) algorithms; 2) utilizing FT as the baseline allows us to explore multiple architectures, and we found them to have a significant impact on down-stream few-shot tasks, even with similar pre-training performances; 3) by decoupling pre-training and few-shot learning, MoFSOD allows us to explore the impact of different pre-training datasets, and the right choice can boost the performance of the down-stream tasks significantly. Based on these findings, we list possible avenues of investigation for improving FSOD performance and propose two simple modifications to existing algorithms that lead to SOTA performance on the MoFSOD benchmark. The code is available here.

Keywords: Few-Shot Learning, Object Detection

1 Introduction

Convolutional neural networks have led to significant progress in object detection by learning with a large number of training images with annotations [40,38,3,4]. However, humans can easily localize and recognize new objects with only a

⋆Work done at AWS. †Corresponding author.

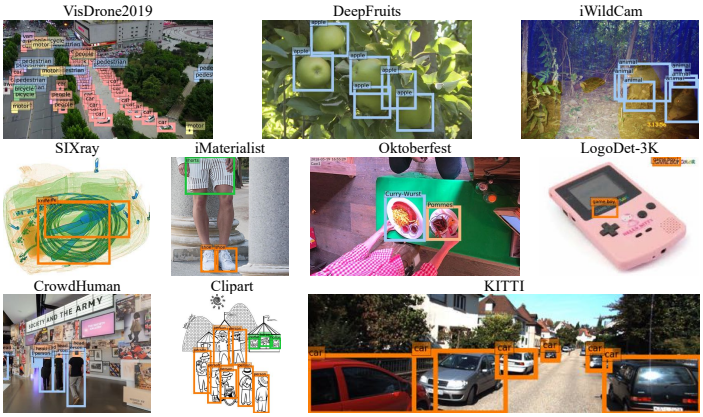
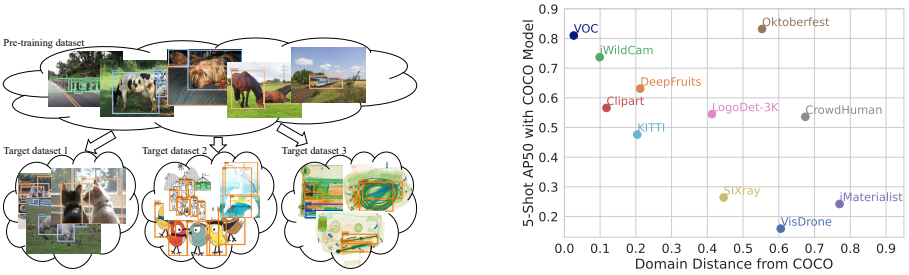


Fig. 1: Sample images in the proposed FSOD benchmark.

few examples. Few-shot object detection (FSOD) is a task to address this setting [26,56,9,47,36]. FSOD is desirable for many real-world applications in diverse domains due to lack of training data, difficulties in annotating them, or both, *e.g.*, identifying new logos, detecting anomalies in the manufacturing process or rare animals in the wild, *etc*. These diverse tasks naturally have vast differences in class distribution and style of images. Moreover, large-scale pre-training datasets in the same domain are not available for many of these tasks. In such cases, we can only rely on existing natural image datasets, such as COCO [31] and OpenImages [28] for pre-training.

Despite the diverse nature of FSOD tasks, FSOD benchmarks used in prior works are limited to a homogeneous setting [26,56,9,47,36], such that the pre-training and few-shot test sets in these benchmarks are from the same domain, or even the same dataset, *e.g.*, VOC [8] $15 + 5$ and COCO [31] $60 + 20$ splits. The class distributions of such few-shot test sets are also fixed to be balanced. While they provide an artificially balanced environment for evaluating different algorithms, it might lead to skewed conclusions for applying them in more realistic scenarios. Note that few-shot classification suffered from the same problem in the past few years [52,39]; Meta-dataset [49] addressed the problem with 10 different domains and a sophisticated scheme to sample imbalanced few-shot episodes.

Inspired by Meta-dataset [49], we propose a Multi-dOmain FSOD (MoFSOD) benchmark consisting of 10 datasets from 10 different domains, as shown in Figure 1. The diversity of MoFSOD datasets can be seen in Figure 2 where the domain distance of each dataset to COCO [31] is depicted. Our benchmark enables us to estimate the performance of FSOD algorithms across domains and settings and helps in better understanding of various factors, such as pre-training, architectures, *etc*., that influence the algorithm performance. In addition, we propose a simple natural K-shot sampling algorithm that encourages more diversity in class distributions than balanced sampling.



(a) Multi-domain datasets.

(b) Domain distance vs 5-shot performance.

Fig. 2: (a) Real-world applications of FSOD are not limited to the natural image domain; we propose to pre-train models on large-scale natural image datasets and transfer to target domains for few-shot learning. (b) We measure the domain distance between datasets (see Section 3.2 for details) in the benchmark and COCO, and plot against 5-shot AP50 of these datasets fine-tuned from a model pre-trained on COCO. VOC is added for reference. The figure shows the benchmark covers a wide range of domains.

Building on our benchmark, we extensively study the impact of freezing parameters, detection architectures, pre-training datasets, and the effectiveness of several state-of-the-art (SOTA) FSOD algorithms [56,47,36]. Our empirical study leads to rethinking conventions in the field and interesting findings that can guide future research on FSOD.

Conventionally, in FSOD or general few-shot learning, it is believed that freezing parameters to avoid overfitting is helpful or even crucial for good performance [56,47,45,58]. If we choose to tune more parameters, specific components or designs must be added, such as weight imprinting [59] or decoupled gradient [36], to prevent overfitting. Our experiments in the MoFSOD show that these design choices might be helpful when pre-training and few-shot learning are in similar domains, as in previous benchmarks. However, if we consider a broader spectrum of domains, unfreezing the whole network results in better overall performance, as the network has more freedom to adapt. We further demonstrate a correlation between the performance gain of tuning more parameters and domain distance (see Figure 3a). Overall, *fine-tuning (FT) is a strong baseline* for FSOD on MoFSOD without any bells and whistles.

Using FT as a baseline allows us to explore the impact of different architectures on FSOD tasks. Previous FSOD methods [56,9,47,36] need to make architecture-specific design choices; hence focus on a single architecture – mostly Faster R-CNN [40], while we conduct extensive study on the impact of different architectures, *e.g.*, recent development of anchor-free [69,71] and transformer-based architectures [4,74], on few-shot performance. Surprisingly, we find that even with similar performance on COCO, different architectures have very different downstream few-shot performances. This finding suggests the potential benefits of specifically designed few-shot architectures for improved performance.

Moreover, unlike previous benchmarks, which split the pre-training and few-shot test sets from the same datasets (VOC or COCO), MoFSOD allows us to freely choose different pre-training datasets and explore the potential benefits of large-scale pre-training. To this end, we systematically study the effect of pre-training datasets with ImageNet [6], COCO [31], LVIS [17], FSOD-Dataset [9], Unified [70], and the integration of large-scale language-vision models. Similar to observations in recent works in image classification [27] and NLP [2,37], we find that large-scale pre-training can play a crucial role for downstream few-shot tasks.

Finally, motivated by the effectiveness of the unfreezing parameters and language-vision pre-training, we propose two extensions: FSCE+ and LVIS+. FSCE+ extends FSCE [47] to fine-tune more parameters with a simplified fully-connected (FC) detection head. LVIS+ follows the idea of using CLIP embedding of class names as the classifier, but instead of using it in zero-shot/open-vocabulary setting as in [68,15], we extend it to few-shot fine-tuning. Both methods achieve SOTA results with/without extra pre-training data.

We summarize our contributions as follows:

- We propose a Multi-dOmain Few-Shot Object Detection (MoFSOD) benchmark to simulate real-world applications in diverse domains.
- We conduct extensive studies on the effect of architectures, pre-training datasets, and hyperparameters with fine-tuning and several SOTA methods on the proposed benchmark. We summarize the observations below:
  - **Unfreezing more layers** do not lead to detrimental overfitting and improve the FSOD performance across different domains.
  - **Object detection model architectures** have a significant impact on the FSOD performance even when the architectures have a similar performance on the pre-training dataset.
  - **Pre-training datasets** play an important role in the downstream FSOD performance. Effective utilization of the pre-training dataset can significantly boost performance.
- Based on these findings, we propose two extensions that outperform SOTA methods by a significant margin on our benchmark.

2 Related Work

Meta-learning-based methods for FSOD are inspired by few-shot classification. Kang et al. [26] proposed a meta feature extractor with feature reweight module, which maps support images to mean features and reweight query features with the mean features, inspired by protoypical networks [45]. Meta R-CNN [63] extended the idea with an extra predictor-head remodeling network to extract class-attentive vectors. MetaDet [58] proposed meta-knowledge transfer with weight prediction module.

Two-stream methods take one query image and support images as inputs, and use the correlations between query and support features as the final features to the detection head and the Region Proposal Network (RPN). Several works in

this direction [9,66,18] have shown competitive results. These methods require all classes to have at least one support image to be fed to the model, which makes the overall process slow.

Fine-tuning-based methods update only the linear classification and regression layers [56], the whole detection head and RPN with an additional contrastive loss [47], or decoupling the gradient of RPN and the detection head while updating the whole network [36]. These methods are simple yet have shown competitive results. We focus on benchmarking them due to their simplicity, efficiency, and higher performance than other types.

Multi-domain few-shot classification benchmarks. In few-shot classification, miniImageNet [52] and tieredImageNet [39] have been used as standard benchmarks. Similar to benchmarks in FSOD, they are divided into two splits and used for pre-training and few-shot learning, respectively, such that they are in the same natural image domain. Recent works have proposed new benchmarks to address this issue: Tseng *et al*. [50] proposed a cross-domain few-shot classification benchmark with five datasets from different domains. Triantafillou *et al*. [49] proposed Meta-Dataset, which is a large-scale few-shot classification benchmark with ten datasets and a sophisticatedly designed sampling algorithm to sample realistically imbalanced few-shot training datasets. Although not specifically catering to few-shot applications, Wang *et al*. [57] proposed universal object detection, which aims to cover multi-domain datasets with a single model for high-shot object detection.

3 MoFSOD: A Multi-Domain FSOD Benchmark

In this section, we first describe existing FSOD benchmarks and their limitations. Then, we propose a Multi-dOmain FSOD (MoFSOD) benchmark.

3.1 Existing Benchmarks and Limitations

Recent FSOD works have evaluated their methods in PASCAL VOC $15 + 5$ and MS COCO $60 + 20$ benchmarks proposed by Kang *et al*. [26]. From the original VOC [8] with 20 classes and COCO [31] with 80 classes, they took 25% of classes as novel classes for few-shot learning, and the rest of them as base classes for pre-training. For VOC $15 + 5$, three splits were made, where each of them consists of 15 base classes for pre-training, and the other 5 novel classes for few-shot learning. For each novel class, $K = \{1, 3, 5, 10\}$ object instances are sampled, which are referred to as shot numbers. For COCO, 20 classes overlapped with VOC are considered novel classes, and $K = \{10, 30\}$-shot settings are used. Different from classification, as an image usually contains multiple annotations in object detection, sampling exactly K annotations per class is difficult. Kang *et al*. [26] proposed pre-defined support sets for few-shot training, which would cause overfitting [23]. Wang *et al*. [56] proposed to sample few-shot training datasets with different random seeds to mitigate this issue, but the resulting

sampled datasets often contain more than K instances. While these benchmarks contributed to the research progress in FSOD, they have several limitations.

First, these benchmarks do not capture the breadth of few-shot tasks and domains as they sample few-shot task instances from a single dataset, as we discussed in Section 1. Second, these benchmarks contain only a fixed number of classes, 5 or 20. However, real-world applications might have a varying number of classes, ranging from one class, *e.g.*, face/pedestrian detection [64,67], to thousands of classes, *e.g.*, logo detection [55]. Last but not least, these benchmarks are constructed with the balanced K-shot sampling. For example, in the 5-shot setting, a set of images containing exactly 5 objects [26] is pre-defined. Such a setting is unlikely in real-world few-shot tasks. We also demonstrate that such a sampling strategy can lead to high variances in the performance of multiple episodes (see Table 2b). Moreover, different from classification, object detection datasets tend to be imbalanced due to the multi-label nature of the datasets. For example, COCO [31] and OpenImages [28] have a more dominant number of person instances than any other objects. The benchmark datasets should also explore these imbalanced scenarios.

3.2 Multi-Domain Benchmark Datasets

While FSOD applications span a wide range of domains, gathering enough pre-training data from these domains might be difficult. Hence, it becomes important to test the few-shot algorithm performance in settings where the pre-training and few-shot domains are different. Similar to Meta-dataset [49] in few-shot classification, we propose to extend the benchmark with datasets from a wide range of domains rather than a subset of natural image datasets. Our proposed benchmark consists of 10 datasets from 10 domains: VisDrone [72] in aerial images, DeepFruits [41] in agriculture, iWildCam [1] in animals in the wild, Clipart in cartoon, iMaterialist [16] in fashion, Oktoberfest [76] in food, LogoDet-3K [55] in logo, CrowdHuman [44] in person, SIXray [34] in security, and KITTI [12] in traffic/autonomous driving. We provide statistics of these datasets in Table 1a. The number of classes varies from 1 to 352, and that of boxes per image varies from 1.2 to 54.4, covering a wide range of scenarios.

In Figure 2b, we illustrate the diversity of domains in our benchmark by computing the domain distances between these datasets and COCO [31] and plotting against the 5-shot performance of fine-tuning (FT) on each dataset from a model pre-trained on COCO. Specifically, we measure the domain similarity by calculating the recall of a pre-trained COCO model on each dataset in a class-agnostic fashion, similar to the measurement of unsupervised object proposals [21]. Intuitively, if a dataset is in a domain similar to COCO, then objects in the dataset are likely to be localized well by the model pre-trained on COCO. As a reference, VOC has a recall of 97%. For presentation purpose, we define $(1 - \text{recall})$ as the domain distance. We can see diverse domain distances in the benchmark, ranging from 0.1 to 0.8. Interestingly, the domain distance also correlates with the FSOD performance. Although this is not the only deciding factor, as the intrinsic properties (such as the similarity between training

Table 1: Statistics of pre-training and MoFSOD datasets (Table 1a) and performance of different architectures on the pre-training datasets (Table 1b).

Domain	Dataset	# classes	# training images
Natural Image	COCO	80	117k
	FSODD	800	56k
	LVIS	1203	100k
	Unified	723	2M

Domain	Dataset	# classes	# bboxes per image
Aerial	VisDrone	10	54.4
Agriculture	DeepFruits	7	5.6
Animal	iWildCam	1	1.5
Cartoon	Clipart	20	3.3
Fashion	iMaterialist	46	7.3
Food	Oktoberfest	15	2.4
Logo	LogoDet-3K	352	1.2
Person	CrowdHuman	2	47.1
Security	SIXray	5	2.1
Traffic	KITTI	4	7.0

(a) **Top:** statistics of pre-training datasets. **Bottom:** statistics of MoFSOD datasets.

Dataset	Architecture	AP
COCO	Faster R-CNN	42.7
	Cascade R-CNN	45.1
	CenterNet2	45.3
	RetinaNet	39.3
	Deformable DETR	46.3
	Cascade R-CNN-P67	45.9
LVIS	Faster R-CNN	24.2
	CenterNet2	28.3
	Cascade R-CNN-P67	26.2

(b) Performance of benchmark architectures pre-trained on COCO and LVIS. FSODD and Unified do not have a pre-defined validation/test set, so we do not measure their pre-training performances.

and test datasets) of a dataset also play an important role, we can still see the linear correlation between the domain distance and 5-shot performance with the Pearson correlation coefficient -0.43. Oktoberfest and CrowdHuman are outliers in our analysis possibly as they are relatively easy.

Natural K-Shot Sampling. We use a natural K-shot sampling algorithm to maintain the original class distribution for this benchmark. Specifically, we sample $C \times K$ images from the original dataset without worrying about class labels, where C is the number of classes of the original dataset. Then, we check missing images to ensure we have at least one image for each class of all classes. We provide the details in Appendix A. The comparison between the balanced K-shot and natural K-shot sampling shows that our conclusions do not change based on the sampling algorithm, but the performance of the natural K-shot sampling is more consistent on different episodes (see Table 2b) and covers imbalanced class distributions existing in the real-world applications.

Evaluation Protocol. To evaluate the scalability of methods, we experiment with four different average shot numbers, $K = \{1, 3, 5, 10\}$. We first sample a few-shot training dataset from the original training dataset with the natural K-shot sampling algorithm for each episode. Then, we initialize the object detection model with pre-trained model parameters and train the model with FSOD methods. For evaluation, we randomly sample 1k images from the original test set if the test set is larger than 1k. We repeat this episode 10 times with different random seeds for all multi-domain datasets and report the average of the mean and standard deviation of the performance.

Metrics. As evaluation metrics, we use AP50 and the average rank among compared methods [49]. AP50 stands for the average precision of predictions where the IoU threshold is 0.5, and the rank is an integer ranging from 1 to the

number of compared methods, where the method with the highest AP50 gets rank 1. We first take the best AP50 among different hyperparameters at the end of training for each episode, compute the mean and standard deviation of AP50 and the rank over 10 episodes, and then average them over different datasets and/or shots, depending on the experiments.

4 Experiments

In this section, we conduct extensive experiments on MoFSOD and discuss the results. For better presentations, we highlight compared methods and architectures in *italics* and pre-training/few-shot datasets in **bold**.

4.1 Experimental Setup

Model architecture. We conduct experiments on six different architectures. For simplicity, we use ResNet-50 [20] as the backbone of all architectures. We also employ deformable convolution v2 [73] in the last three stages of the backbone. Specifically, we benchmark two-stage object detection architectures: 1) *Faster R-CNN* [40] 2) *Cascade R-CNN* [3], and the newly proposed 3) *CenterNet2* [69], and one-stage architectures: 4) *RetinaNet* [30], as well as transformer-based 5) *Deformable-DETR* [74]. Note that all architectures utilize Feature Pyramid Networks (FPN) [29] or similar multi-scale feature aggregation techniques. In addition, we also experiment the combination of the FPN-P67 design from *RetinaNet* and *Cascade R-CNN*, dubbed 6) *Cascade R-CNN-P67* [70]. We conduct our architecture analysis pre-trained on **COCO** [31] and **LVIS** [17]. Table 1b summarizes the architectures and their pre-training performance.

Freezing parameters. Based on the design of detectors, we can think of three different levels of fine-tuning the network: 1) only the last classification and regression fully-connected (FC) layer [56], 2) the detection head consisting of several FC and/or convolutional layers [9], and 3) the whole network, *i.e.*, standard fine-tuning.[1] We study the effects of these three ways of tuning on different domains in MoFSOD with *Faster R-CNN* [40].

Pre-training datasets. To explore the effect of pre-training dataset, we conduct experiments on five pre-training datasets: **ImageNet**[2] [6], **COCO** [31], **FSODD**[3] [9], **LVIS** [17], and **Unified**, which is a union of OpenImages v5 [28], Object365 v1 [43], Mapillary [7], and **COCO**, combined as in [70]. To reduce the combinations of different architectures and pre-training datasets, we conduct most of the studies on the best performing architecture *Cascade R-CNN-P67*.

[1] When training an object detection model, the batch normalization layers [25] and the first two macro blocks of the backbone (`stem` and `res2`) are usually frozen, even for large-scale datasets. We follow this convention in our paper.

[2] This is ImageNet-1K for classification, which is commonly used for pre-training standard object detection methods, *i.e.*, we omit pre-training on an object detection task.

[3] The name of the dataset is also FSOD, so we introduce an additional D to distinguish the dataset from the task.

Also, inspired by [68,15], we experiment with the effect of CLIP [37] embeddings to initialize the final classification layer of detector when pre-training on **LVIS** dubbed **LVIS+**. In addition, **LVIS++** uses the backbone pre-trained on the ImageNet-21K classification task instead of ImageNet-1K before pre-training on **LVIS**. All experiments are done with Detectron2 [60].

Hyperparameters. For pre-training, we mostly follow standard hyperparameters of the corresponding method, with the addition of deformable convolution v2 [73]. On **COCO** and **LVIS**, for *Faster R-CNN*, *Cascade R-CNN*, and *Cascade R-CNN-P67*, we use the 3× scheduler with 270k iterations, the batch size of 16, the SGD optimizer with initial learning rate of 0.02 decaying by the factor of 0.1 at 210k and 250k. For *RetinaNet*, the initial learning rate is 0.01 [30]. For *CenterNet2*, following [69], we use the *CenterNet* [71] as the first stage and the Cascade R-CNN head as the second stage, where the other hyperparameters are the same as above. For *Deformable-DETR* [74], we follow the two-stage training of 50 epochs, the AdamW optimizer, and the initial learning rate of 0.0002 decaying by the factor of 0.1 at 40 epochs. On **FSODD**, we train for 60 epochs with the learning rate of 0.02 decaying by the factor of 0.1 at 40 and 54 epochs and the batch size of 32. On **Unified**, following [70], the label space of four datasets are unified, the dataset-aware sampling and equalization loss [48] are applied to handle long-tailed distributions, and training is done for 600k iterations with the learning rate of 0.02 decaying by the factor of 0.1 at 400k and 540k iterations, and the batch size of 32.

For few-shot training, we train models for 2k iterations with the batch size of 4 on a single V100 GPU.[4] For *Faster R-CNN*, *Cascade R-CNN*, *Cascade R-CNN-P67*, and *CenterNet2*, we train models with the SGD optimizer and different initial learning rates in {0.0025, 0.005, 0.01} and choose the best, where the learning rate is decayed by the factor of 0.1 after 80% of training. For *RetinaNet*, we halve the learning rates to {0.001, 0.0025, 0.005}, as we often observe training diverges with the learning rate of 0.01. For *Deformable-DETR* and *CenterNet2 with CLIP*, we use the AdamW optimizer and initial learning rates in {0.0001, 0.0002, 0.0004}.[5]

Compared methods. *TFA [56]* or *Two-stage Fine-tuning Approach* has shown to be a simple yet effective method for FSOD. TFA fine-tunes the box regressor and classifier on the few-shot dataset while freezing the other parameters of the model. For this method, we use the FC head, such that TFA is essentially the same as tuning the final FC layer.[6]

FSCE [47] or *Few-Shot object detection via Contrastive proposals Encoding* improves TFA by 1) additionally unfreezing detection head in the setting of TFA, 2) doubling the number of proposals kept after NMS and halving the number of sampled proposals in the RoI head, and 3) optimizing the contrastive proposal

[4] The batch size could be less than 4 if the sampled dataset size is less than 4, *e.g.*, when the number of classes and shot number K is 1, and the batch size has to be 1.
[5] The learning rates are chosen from our initial experiments on three datasets. Note that training *CenterNet2* with AdamW results in worse performance than SGD.
[6] Replacing the FC head with the cosine-similarity results in a similar performance.

Table 2: The K-shot performance on MoFSOD with $K = \{1, 3, 5, 10\}$.

Method	Pre-training	1-shot	3-shot	5-shot	10-shot
TFA [56]	COCO	23.4 ± 4.6	29.2 ± 1.8	32.0 ± 1.3	35.2 ± 1.2
FSCE-base [47]		30.5 ± 5.3	39.4 ± 1.6	43.9 ± 1.1	50.2 ± 1.1
FSCE-con [47]		29.4 ± 2.5	38.8 ± 1.5	43.6 ± 1.2	50.4 ± 1.0
DeFRCN [36]		29.3 ± 4.2	37.8 ± 3.1	41.6 ± 1.9	48.2 ± 1.9
Ours-FT		**31.5** ± 2.0	41.1 ± 1.8	46.1 ± 1.4	52.6 ± 1.8
Ours-FSCE+		31.2 ± 2.4	**41.3** ± 1.5	**46.4** ± 1.1	**53.2** ± 1.5
Ours-FT+	COCO	35.4 ± 1.8	44.7 ± 1.6	49.9 ± 1.2	56.4 ± 1.1
Ours-FT++	LVIS++	**35.8** ± 3.4	**47.2** ± 2.1	**52.6** ± 1.4	**59.4** ± 1.1

(a) Performance with the natural K-shot.

Method	Pre-training	1-shot	3-shot	5-shot	10-shot
TFA [56]	COCO	22.9 ± 5.4	27.6 ± 2.1	28.3 ± 5.9	31.6 ± 2.7
FSCE-con [47]		26.8 ± 3.9	33.6 ± 4.9	36.3 ± 5.0	40.2 ± 5.4
DeFRCN [36]		27.5 ± 5.1	34.8 ± 2.2	37.4 ± 1.7	40.1 ± 6.5
Ours-FT		**28.8** ± 2.3	34.8 ± 3.1	37.4 ± 4.4	42.1 ± 5.4
Ours-FSCE+		28.7 ± 3.8	**36.5** ± 5.3	**38.2** ± 5.0	**42.7** ± 5.7

(b) Performance with the balanced K-shot.

Architecture	Pre-training	1-shot	3-shot	5-shot	10-shot
Faster R-CNN	COCO	31.5 ± 2.0	41.1 ± 1.8	46.1 ± 1.4	52.6 ± 1.8
Cascade R-CNN		31.5 ± 2.1	41.2 ± 1.5	45.6 ± 1.4	52.7 ± 1.2
CenterNet2		29.1 ± 5.2	40.2 ± 1.9	45.3 ± 1.8	52.5 ± 1.9
RetinaNet		25.4 ± 4.0	34.8 ± 2.2	40.6 ± 1.7	48.8 ± 1.2
Deformable-DETR		32.0 ± 2.9	42.3 ± 1.6	47.4 ± 1.4	54.7 ± 1.0
Cascade R-CNN-P67		**35.4** ± 1.8	**44.7** ± 1.6	**49.9** ± 1.2	**56.4** ± 1.1
Faster R-CNN	LVIS	31.7 ± 1.7	41.6 ± 1.5	46.4 ± 1.2	53.6 ± 1.0
CenterNet2		28.1 ± 3.3	39.0 ± 1.7	44.3 ± 1.2	51.6 ± 1.0
Cascade R-CNN-P67		**34.4** ± 1.2	**44.0** ± 1.6	**48.7** ± 1.4	**55.6** ± 1.0

(c) The effect of different architectures.

Architecture	Pre-training	1-shot	3-shot	5-shot	10-shot
Cascade R-CNN-P67	ImageNet	13.5 ± 1.6	23.2 ± 1.5	29.5 ± 1.2	37.7 ± 1.2
	COCO	**35.4** ± 1.8	44.7 ± 1.6	49.9 ± 1.2	56.4 ± 1.1
	FSODD	26.7 ± 2.9	36.9 ± 1.5	42.3 ± 1.2	49.1 ± 1.0
	LVIS	34.4 ± 2.0	44.0 ± 1.6	48.7 ± 1.4	55.6 ± 1.0
	Unified	33.3 ± 2.2	44.3 ± 1.4	49.6 ± 1.2	56.6 ± 1.1
	LVIS+	34.7 ± 4.2	**46.6** ± 1.6	**52.0** ± 1.0	**59.0** ± 1.0
CenterNet2	COCO	29.1 ± 5.2	40.2 ± 1.9	45.3 ± 1.8	52.5 ± 1.9
	LVIS	28.1 ± 3.3	39.0 ± 1.7	44.3 ± 1.2	51.6 ± 1.0
	LVIS+	34.9 ± 3.2	46.5 ± 1.9	51.8 ± 1.3	58.8 ± 1.0
	LVIS++	**35.8** ± 3.4	**47.2** ± 2.1	**52.6** ± 1.4	**59.4** ± 1.1

(d) The effect of different pre-training.

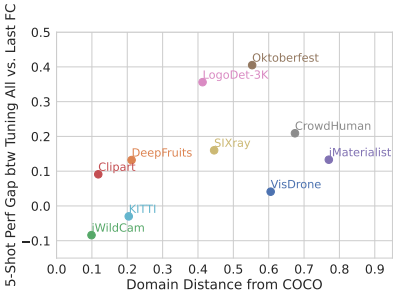
encoding loss. While the original work did not apply the contrastive loss for extremely few-shot settings (less than 3), we explicitly compare two versions in all shots: without (*FSCE-base*, the same as tuning the detection head) and with the contrastive loss (*FSCE-con*).

DeFRCN [36] or *Decoupled Faster R-CNN* can be distinguished with other methods by 1) freezing only the R-CNN head, 2) decoupling gradients to suppress gradients from RPN while scaling those from the R-CNN head, and 3) calibrating the classification score from an offline prototypical calibration block (PCB), which is a CNN-based prototype classifier pre-trained on ImageNet [6]. We note that PCB does not re-scale input images in their original implementation, unlike the object detector, so we manually scaled images to avoid GPU memory overflow if they are too large.

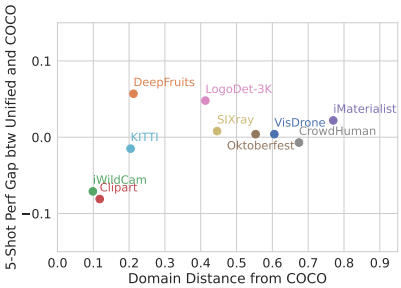
FT or *Fine-tuning* does not freeze model parameters as done for other methods. Though it is undervalued in prior works, we found that this simple baseline outperforms state-of-the-art methods in our proposed benchmark. All experiments are done with this method unless otherwise specified.

4.2 Experimental Results and Discussions

Effect of tuning more parameters. We first analyze the effect of tuning more or fewer parameters on MoFSOD. In Table 2a and Table 3a, we examine three methods freezing different number of parameters when fine-tuning: *TFA* as tuning the last FC layers only, *FSCE-base* as tuning the detection head, and *FT* as tuning the whole network. We observe that freezing fewer parameters improves the average performance: tuning the whole network (*FT*) shows better performance than others, while tuning the last FC layers only (*TFA*) shows lower performance than others. Also, the performance gap becomes larger as the size of few-shot training datasets increases. For example, *FT* outperforms



(a) Performance gain by tuning *the whole network FT*) rather than *the last FC layer only* (*TFA*) vs. domain distance.



(b) Performance gap between **Unified** and **COCO** vs. domain distance.

Fig. 3: We demonstrate the correlation between tuning more parameters and domain distance in Figure 3a and the correlation between pre-training datasets and domain distance in Figure 3b.

FSCE-base and *TFA* by 1.0% and 8.1% in 1-shot, and 2.4%, 17.4% in 10-shot, respectively. This contrasts with the conventional belief that freezing most of the parameters generally improves the performance of few-shot learning, as it prevents overfitting [45,10,56,47]. However, this is not necessarily true for FSOD. For example, in the standard two-stage object detector training, RPN is class-agnostic, such that its initialization for training downstream few-shot tasks can be the one pre-trained on large-scale datasets, preserving the pre-trained knowledge on objectness. Also, the detection head utilizes thousands of examples even in few-shot scenarios, because RPN could generate 1–2k proposals per image. Hence, the risk of overfitting is relatively low.

However, fine-tuning more parameters does not always improve performance. Figure 3a illustrates the performance gain by tuning more parameters with respect to the domain distance. There is a linear correlation, *i.e.*, the performance gain by fine-tuning more parameters increases when the domain distance increases. This implies that fine-tuning fewer parameters to preserve the pre-trained knowledge is better when the the few-shot dataset is close to the pre-training dataset. Hence, for datasets close to **COCO**, such as **KITTI** and **iWildCam**, *tuning Last FC layers* (*TFA*) is the best performing method. From these observations, an interesting research direction might be exploring a sophisticated tuning of layers based on the few-shot problem definition and the domain gap between pre-training and few-shot tasks.

One way to design such sophisticated tuning is to develop a better measure for domain distance. In fact, the proposed measure with class-agnostic object recall has limitations. If we decouple the object detection task into localization (background vs. foreground) and classification (among foreground classes), then the proposed domain distance is biased towards measuring localization gaps. Therefore, it ignores the potential classification gaps that would also require tuning more layers. For example, although we can get a good coverage with the propos-

Table 3: Per-dataset 5-shot performance of the effects of tuning different parameters, different architectures and pre-training datasets.

5-shot	Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Unfrozen	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Last FC layers (TFA [56])	10.1 ± 0.6	47.5 ± 1.8	**71.7** ± 2.4	40.2 ± 2.6	8.0 ± 0.9	41.1 ± 4.8	14.2 ± 3.7	30.6 ± 0.8	7.9 ± 2.0	**48.3** ± 3.4	32.0 ± 1.3	2.7 ± 0.2
Detection head (FSCE-base [47])	13.0 ± 0.7	59.2 ± 3.2	70.6 ± 1.7	43.5 ± 2.5	20.5 ± 0.9	70.7 ± 3.5	47.2 ± 3.5	**51.6** ± 2.0	15.9 ± 2.2	47.1 ± 4.3	43.9 ± 1.1	1.9 ± 0.2
Whole network (Ours-FT)	**14.2** ± 0.8	**60.7** ± 3.8	63.3 ± 5.7	**49.3** ± 3.5	**21.3** ± 0.8	**81.6** ± 4.0	**49.8** ± 3.5	51.5 ± 2.2	**23.9** ± 3.9	45.3 ± 3.7	**46.1** ± 1.4	**1.5** ± 0.2

(a) Fine-tuning different number of parameters with Faster R-CNN pre-trained on COCO.

5-shot		Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Architecture	Pre-training	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Faster R-CNN	COCO	14.2 ± 0.8	60.7 ± 3.8	63.3 ± 5.7	49.3 ± 3.5	21.3 ± 0.8	81.6 ± 4.0	49.8 ± 3.5	51.5 ± 2.2	23.9 ± 3.9	45.3 ± 3.7	46.1 ± 1.4	3.4 ± 0.3
Cascade R-CNN	COCO	13.0 ± 0.8	58.9 ± 3.3	66.8 ± 3.7	50.8 ± 2.8	20.5 ± 0.6	80.5 ± 2.2	48.5 ± 4.7	51.4 ± 2.0	21.6 ± 4.4	44.4 ± 4.1	45.6 ± 1.4	4.1 ± 0.4
CenterNet2	COCO	13.5 ± 0.7	59.0 ± 4.7	61.6 ± 4.7	49.2 ± 7.6	22.2 ± 1.9	79.7 ± 3.5	51.0 ± 4.2	51.4 ± 3.7	22.2 ± 4.8	43.7 ± 4.9	45.3 ± 1.8	3.9 ± 0.3
RetinaNet	COCO	9.9 ± 0.6	55.8 ± 2.7	59.2 ± 6.8	26.8 ± 2.2	17.6 ± 0.4	79.5 ± 4.0	49.2 ± 3.5	47.6 ± 1.9	20.6 ± 3.3	39.7 ± 3.3	40.6 ± 1.7	5.4 ± 0.5
Deformable-DETR	COCO	15.0 ± 0.6	**68.8** ± 4.3	66.0 ± 4.3	43.7 ± 1.6	22.4 ± 1.1	77.2 ± 4.6	50.3 ± 3.5	**56.7** ± 2.4	26.3 ± 3.8	**47.9** ± 3.2	47.4 ± 1.4	2.6 ± 0.5
Cascade R-CNN-P67	COCO	**15.9** ± 0.8	63.1 ± 2.3	**73.7** ± 2.8	**56.6** ± 2.3	**24.2** ± 0.8	**83.2** ± 2.4	**54.5** ± 4.4	53.6 ± 1.8	**26.4** ± 3.9	47.6 ± 3.6	**49.9** ± 1.2	**1.5** ± 0.3
Faster R-CNN	LVIS	14.2 ± 0.7	**66.2** ± 3.4	69.3 ± 3.7	39.8 ± 1.8	28.0 ± 0.6	80.7 ± 2.7	51.3 ± 4.2	48.3 ± 2.2	24.7 ± 3.6	41.3 ± 3.5	46.4 ± 1.2	2.1 ± 0.3
CenterNet2	LVIS	13.3 ± 0.7	64.6 ± 3.5	50.2 ± 25.3	34.1 ± 2.5	25.6 ± 0.5	81.6 ± 2.8	53.7 ± 3.8	46.6 ± 2.0	22.8 ± 3.3	38.9 ± 3.9	43.1 ± 6.9	2.7 ± 0.3
Cascade R-CNN-P67	LVIS	**15.2** ± 0.7	65.4 ± 2.0	**71.7** ± 2.0	**46.0** ± 2.5	**30.2** ± 0.6	**81.9** ± 3.0	**56.6** ± 5.0	**49.6** ± 1.7	25.9 ± 4.3	**44.7** ± 3.7	**48.7** ± 1.4	**1.2** ± 0.3

(b) Performance of different architectures pre-trained on COCO and LVIS.

5-shot		Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Architecture	Pre-training	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Cascade R-CNN-P67	ImageNet	9.8 ± 0.5	53.0 ± 3.5	7.4 ± 3.2	13.1 ± 2.5	19.2 ± 0.6	77.4 ± 3.4	46.4 ± 4.4	32.0 ± 3.1	13.1 ± 3.2	23.8 ± 3.4	29.5 ± 1.2	6.0 ± 0.0
	COCO	15.9 ± 0.8	63.1 ± 2.3	**73.7** ± 2.8	**56.6** ± 2.3	24.2 ± 0.8	83.2 ± 2.4	54.5 ± 4.4	**53.6** ± 1.8	26.4 ± 3.9	47.6 ± 3.6	49.9 ± 1.2	2.8 ± 0.4
	FSODD	10.6 ± 0.6	67.5 ± 3.7	64.5 ± 3.4	26.5 ± 2.0	21.2 ± 0.5	82.9 ± 2.9	55.7 ± 3.8	38.4 ± 2.2	23.2 ± 3.7	32.3 ± 3.6	42.3 ± 1.2	4.3 ± 0.4
	LVIS	15.2 ± 0.7	65.4 ± 2.0	71.7 ± 2.0	46.0 ± 2.5	30.2 ± 0.6	81.9 ± 3.0	56.6 ± 5.0	49.6 ± 1.7	25.9 ± 4.3	44.7 ± 3.7	48.7 ± 1.4	3.4 ± 0.4
	Unified	16.3 ± 0.9	68.8 ± 3.0	66.6 ± 4.0	48.5 ± 2.9	26.4 ± 0.7	**83.6** ± 2.9	59.3 ± 3.7	52.9 ± 1.6	27.2 ± 3.6	46.1 ± 3.9	49.6 ± 1.2	2.5 ± 0.4
	LVIS+	**18.5** ± 1.0	**76.8** ± 3.3	59.5 ± 4.1	52.6 ± 2.0	**31.5** ± 0.7	82.5 ± 3.3	**61.2** ± 2.7	52.3 ± 1.9	**36.0** ± 2.5	**49.6** ± 3.9	**52.0** ± 1.1	**1.9** ± 0.5
CenterNet2	COCO	13.5 ± 0.7	59.0 ± 4.7	61.6 ± 4.7	49.2 ± 7.6	22.2 ± 1.9	79.7 ± 3.5	51.0 ± 4.2	51.4 ± 3.7	22.2 ± 4.8	43.7 ± 4.9	45.3 ± 1.8	3.2 ± 0.3
	LVIS	13.3 ± 0.7	64.6 ± 3.5	61.5 ± 4.3	34.1 ± 2.5	25.6 ± 0.5	**81.6** ± 2.8	53.7 ± 3.8	46.6 ± 2.0	22.8 ± 3.3	38.9 ± 3.9	44.3 ± 1.2	3.2 ± 0.2
	LVIS+	**18.2** ± 0.9	74.0 ± 3.3	63.7 ± 3.8	50.5 ± 1.8	31.5 ± 0.8	80.4 ± 4.0	**62.3** ± 3.2	**54.0** ± 2.1	37.3 ± 4.1	46.5 ± 4.7	51.8 ± 1.3	2.0 ± 0.4
	LVIS++	18.1 ± 0.9	**77.5** ± 2.4	**64.4** ± 4.7	**52.1** ± 1.8	**33.1** ± 0.7	80.4 ± 3.5	61.0 ± 4.5	54.7 ± 2.0	**37.8** ± 3.2	**46.9** ± 4.1	**52.6** ± 1.4	**1.5** ± 0.3

(c) Performance of Cascade R-CNN-P67 and CenterNet2 pre-trained on different datasets.

als of **COCO** pre-training for **Clipart** (classic domain adaption dataset) and **DeepFruits** (infrared images), resulting in relatively small domain distances, there exist significant gaps of feature discrimination for fine-grained classification. In this case, we need to tune more parameters for better performance.

Effect of model architectures. A benefit of using *FT* as the baseline is that we can systematically study the effects of model architectures without the constraints of specifically designed components. Many different architectures have been proposed to solve object detection problems; each has its own merits and drawbacks. One-stage methods, such as YOLO [38] and RetinaNet [30], are known for their fast inference speed. However, different from two-stage methods, they do not have the benefit of inheriting the pre-trained class-agnostic RPN. Specifically, the classification layers for discriminating background/foreground and foreground classes need to be reinitialized, as they are often tied together in one-stage methods. We validate this hypothesis by comparing with two-stage methods in Table 2c. Compared to *Faster R-CNN*, *RetinaNet* has 4–6% low performance on MoFSOD. Per-dataset performance in Table 3b shows that *RetinaNet* is worse in all cases.

The same principle of preserving as much information as possible from pre-training also applies for two-stage methods, *i.e.*, reducing the number of randomly reinitialized parameters is better. Specifically, we look into the performance of *Cascade R-CNN* vs *Faster R-CNN*. For *Cascade R-CNN*, we need to

reinitialize and learn three FC layers as there are three stages in the cascade detection head, while we only need to reinitialize the last FC layer for *Faster R-CNN*. However, *Cascade R-CNN* is known to have better performance, as demonstrated in Table 1b. In FSOD, these two factors cancelled out, such that their performance is on a par with each other as shown in Table 2c.

Based on these insights, we extend *Cascade R-CNN* by applying the FPN-P67 architecture [30], similar to [70,69]. Specifically, assuming ResNet-like architecture [20], we use the last three stages of the backbone, namely [`res3`, `res4`, `res5`], instead of the last four in standard FPN. Then, we add P6 and P7 of the FPN features from P5 with two different FC layers, such that RPN takes in features from [`P3`, `P4`, `P5`, `P6`, `P7`] to improve the class-agnostic coverage, which can be inherited for down-stream tasks. While the detection head still uses [`P3`, `P4`, `P5`] only. As shown in Table 2c, the resulting architecture, *Cascade R-CNN-P67* improves *Cascade R-CNN* by 3–4% on the downstream few-shot tasks.

Moreover, recent works proposed new directions of improvement, such as utilizing point-based predictions [71,69] or transformer-based set predictions [4,74]. These methods are unknown quantities in FSOD, as no previous FSOD work has studied them. In our experiments, while *CenterNet2* [69] outperforms *Faster R-CNN* on **COCO** by 2.6% as shown in Table 1b, its FSOD performance on MoFSOD is lower, *e.g.*, 2.4% in 1-shot. In the case of *Deformable-DETR* [74], it outperforms *Faster R-CNN* in both pre-training and few-shot learning, by 3.6% on **COCO** and 2.1% on MoFSOD in 10-shot. These results show that the upstream performance might not necessarily translate to the downstream FSOD performance. We note that we could not observe a significant correlation between the performance gap of different architectures and domain distances.

Effect of pre-training datasets. MoFSOD consists of datasets from a wide range of domains, allowing us to freely explore different pre-training datasets while ensuring domain shifts between pre-training and few-shot learning. We examine the impact of the pre-training datasets with the best performing *Cascade R-CNN-P67* architecture.

In Table 2d, we first observe that pre-training on **ImageNet** for classification results in low performance, as it does not provide a good initialization for downstream FSOD, especially for RPN. On the other hand, compared to **COCO**, **FSODD**, **LVIS**, and **Unified** have more classes and/or more annotations, while they have a fewer, similar, and more number of images, respectively. Pre-training on these larger object detection datasets does not improve the FSOD performance significantly, as shown in Table 2d. For example, pre-training on **Unified** improves the performance over **COCO** when the domain distance from **COCO** is large, such as **Deepfruits** and **LogoDet-3K** as shown in Figure 3b. However, pre-training on **Unified** results in lower performance for few-shot datasets close to **COCO**, such that the overall performance is similar. We hypothesize that this could be due to the non-optimal pre-training of **LVIS** and **Unified**, as these two datasets are highly imbalanced and difficult to train. It could also be the case that even **LVIS** and **Unified** do not have better coverage for these datasets.

On the other end of the spectrum, we can combine the idea of preserving knowledge and large-scale pre-training by utilizing a large-scale language-vision model. Following [15,68], we use CLIP to extract text features from each class name and build a classifier initialized with the text features. In this way, we initialize the classifier with the CLIP text embeddings for downstream few-shot tasks, such that it has strong built-in knowledge of text-image alignment, better than random initialization. As demonstrated in Table 2d, For **LVIS+**, we can see this improves performance significantly by 7.5% for *CenterNet2*, and 3.3% for *Cascade R-CNN-P67* in 5-shot. **LVIS++** pre-trains the backbone on ImageNet-21K instead of ImageNet-1K (before pre-training on **LVIS**), and it further improves over **LVIS+** by 0.8% in 5-shot. However, the benefit of CLIP initialization is valid only when class names are matched with texts presented in CLIP; an exceptional case is **Oktoberfest**, which has German class names, such that **LVIS+** does not help.

Comparison with SOTA methods. Table 2a and Table 2b compare SOTA methods and our proposed methods. For balanced K-shot sampling, we follow Wang *et al*. [56] to sample K instances for each class whenever possible greedily. Here *FT* and *FSCE+* employ a similar backbone/architecture and pre-trained data as all SOTA methods for a fair comparison. We confirm that *FT* is indeed a strong baseline, such that it performs better than other SOTA methods in both natural K-shot and balanced K-shot settings. In addition to *FT*, based on the insights above, we propose several extensions: 1) *FSCE+* is an extension of *FSCE* by tuning the whole network parameters, similar to *FT*. We keep the contrastive proposal encoding loss, but we simplify the classification head from the cosine similarity head to the FC head. We can see the improvement by 2–3% compared to *FSCE* for both natural K-shot and balanced K-shot scenarios and performs slightly better than *FT*. 2) *FT+* replaces *Faster R-CNN* with *Cascade R-CNN-P67*, and it improves over *FT* by 3–4% without sacrificing inference speed or memory consumption much. 3) *FT++* replaces *Faster R-CNN* with *CenterNet2* and uses **LVIS++** for initialization, and it further improves the performance by around 3% in 3-, 5-, and 10-shot. We also observe that while the overall trend of performance is similar for both natural and balanced K-shot sampling, the standard deviation of the natural K-shot performance is less than that of the balanced K-shot.

5 Conclusion

We present the Multi-dOmain Few-Shot Object Detection (MoFSOD) benchmark consisting of 10 datasets from different domains to evaluate FSOD methods. Under the proposed benchmark, we conducted extensive experiments on the impact of freezing parameters, different architectures, and different pre-training datasets. Based on our findings, we proposed simple extensions of the existing methods and achieved state-of-the-art results on the benchmark. In the future, we would like to go beyond empirical studies and modifications, to designing architectures and smart-tuning methods for a wide range of FSOD tasks.

References

1. Beery, S., Agarwal, A., Cole, E., Birodkar, V.: The iwildcam 2021 competition dataset. arXiv preprint arXiv:2105.03494 (2021) 6, 20
2. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: NeurIPS (2020) 4
3. Cai, Z., Vasconcelos, N.: Cascade R-CNN: Delving into high quality object detection. In: CVPR (2018) 1, 8
4. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020) 1, 3, 13
5. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016) 21
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: CVPR. pp. 248–255. Ieee (2009) 4, 8, 10
7. Ertler, C., Mislej, J., Ollmann, T., Porzi, L., Kuang, Y.: Traffic sign detection and classification around the world. arXiv preprint arXiv:1909.04422 (2019) 8
8. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. IJCV **88**(2), 303–338 (2010) 2, 5
9. Fan, Q., Zhuo, W., Tang, C.K., Tai, Y.W.: Few-shot object detection with attention-rpn and multi-relation detector. In: CVPR (2020) 2, 3, 4, 5, 8
10. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML (2017) 11
11. Ge, Y., Zhang, R., Wang, X., Tang, X., Luo, P.: DeepFashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. In: CVPR (2019) 20
12. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012) 6, 21
13. Georgakis, G., Reza, M.A., Mousavian, A., Le, P.H., Košecká, J.: Multiview rgb-d dataset for object instance detection. In: 3DV (2016) 20
14. Goldman, E., Herzig, R., Eisenschtat, A., Goldberger, J., Hassner, T.: Precise detection in densely packed scenes. In: CVPR (2019) 20
15. Gu, X., Lin, T., Kuo, W., Cui, Y.: Open-vocabulary object detection via vision and language knowledge distillation. In: ICLR (2022) 4, 9, 14
16. Guo, S., Huang, W., Zhang, X., Srikhanta, P., Cui, Y., Li, Y., R.Scott, M., Adam, H., Belongie, S.: The imaterialist fashion attribute dataset. arXiv preprint arXiv:1906.05750 (2019) 6, 20
17. Gupta, A., Dollár, P., Girshick, R.B.: LVIS: A dataset for large vocabulary instance segmentation. In: CVPR (2019) 4, 8
18. Han, G., He, Y., Huang, S., Ma, J., Chang, S.F.: Query adaptive few-shot object detection with heterogeneous graph convolutional networks. In: ICCV (2021) 5
19. Häni, N., Roy, P., Isler, V.: MinneApple: a benchmark dataset for apple detection and segmentation. IEEE Robotics and Automation Letters **5**(2), 852–858 (2020) 20
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 8, 13

21. Hosang, J.H., Benenson, R., Schiele, B.: How good are detection proposals, really? In: Valstar, M.F., French, A.P., Pridmore, T.P. (eds.) BMVC (2014) 6
22. Hsieh, M.R., Lin, Y.L., Hsu, W.H.: Drone-based object counting by spatially regularized regional proposal network. In: ICCV (2017) 20
23. Huang, G., Laradji, I., Vazquez, D., Lacoste-Julien, S., Rodriguez, P.: A survey of self-supervised and few-shot object detection. arXiv preprint arXiv:2110.14711 (2021) 5
24. Inoue, N., Furuta, R., Yamasaki, T., Aizawa, K.: Cross-domain weakly-supervised object detection through progressive domain adaptation. In: CVPR (2018) 20
25. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015) 8
26. Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., Darrell, T.: Few-shot object detection via feature reweighting. In: ICCV (2019) 2, 4, 5, 6, 19
27. Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., Houlsby, N.: Big transfer (bit): General visual representation learning. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J. (eds.) ECCV (2020) 4
28. Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Kamali, S., Malloci, M., Pont-Tuset, J., Veit, A., Belongie, S., Gomes, V., Gupta, A., Sun, C., Chechik, G., Cai, D., Feng, Z., Narayanan, D., Murphy, K.: OpenImages: A public dataset for large-scale multi-label and multi-class image classification. Dataset available from https://storage.googleapis.com/openimages/web/index.html (2017) 2, 6, 8
29. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017) 8
30. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017) 8, 9, 12, 13
31. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft COCO: Common objects in context. arXiv:1405.0312 (2014) 2, 4, 5, 6, 8
32. Liu, C., Li, H., Wang, S., Zhu, M., Wang, D., Fan, X., Wang, Z.: A dataset and benchmark of underwater object detection for robot picking. In: 2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW) (2021) 21
33. Liu, J., Lian, J., Yu, Y.: Chestx-det10: Chest x-ray dataset on detection of thoracic abnormalities. arXiv preprint arXiv:2006.10550 (2020) 21
34. Miao, C., Xie, L., Wan, F., Su, C., Liu, H., Jiao, J., Ye, Q.: Sixray: A large-scale security inspection x-ray benchmark for prohibited item discovery in overlapping images. In: CVPR (2019) 6, 21
35. Mogelmose, A., Trivedi, M.M., Moeslund, T.B.: Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. IEEE Transactions on Intelligent Transportation Systems **13**(4), 1484–1497 (2012) 21
36. Qiao, L., Zhao, Y., Li, Z., Qiu, X., Wu, J., Zhang, C.: DeFRCN: Decoupled faster r-cnn for few-shot object detection. In: ICCV (2021) 2, 3, 5, 10, 26
37. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: ICML (2021) 4, 9
38. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: CVPR. pp. 7263–7271 (2017) 1, 12
39. Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J.B., Larochelle, H., Zemel, R.S.: Meta-learning for semi-supervised few-shot classification. In: ICLR (2018) 2, 5

40. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. NeurIPS (2015) 1, 3, 8
41. Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., McCool, C.: DeepFruits: A fruit detection system using deep neural networks. sensors **16**(8), 1222 (2016) 6, 20
42. sgrpanchal31: table-detection-dataset. https://github.com/sgrpanchal31/table-detection-dataset (2018) 21
43. Shao, S., Li, Z., Zhang, T., Peng, C., Yu, G., Zhang, X., Li, J., Sun, J.: Objects365: A large-scale, high-quality dataset for object detection. In: ICCV (2019) 8
44. Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., Sun, J.: CrowdHuman: A benchmark for detecting human in a crowd. arXiv preprint arXiv:1805.00123 (2018) 6, 21
45. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. In: NeurIPS (2017) 3, 4, 11
46. Su, H., Zhu, X., Gong, S.: Open logo detection challenge. In: BMVC (2018) 20
47. Sun, B., Li, B., Cai, S., Yuan, Y., Zhang, C.: Fsce: Few-shot object detection via contrastive proposal encoding. In: CVPR (2021) 2, 3, 4, 5, 9, 10, 11, 12, 19, 22, 23, 24, 26
48. Tan, J., Wang, C., Li, B., Li, Q., Ouyang, W., Yin, C., Yan, J.: Equalization loss for long-tailed object recognition. In: CVPR (2020) 9
49. Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P.A., et al.: Meta-dataset: A dataset of datasets for learning to learn from few examples. In: ICLR (2020) 2, 5, 6, 7
50. Tseng, H.Y., Lee, H.Y., Huang, J.B., Yang, M.H.: Cross-domain few-shot classification via learned feature-wise transformation. In: ICLR (2020) 5
51. Veit, A., Matera, T., Neumann, L., Matas, J., Belongie, S.: Coco-text: Dataset and benchmark for text detection and recognition in natural images. arXiv preprint arXiv:1601.07140 (2016) 21
52. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: NeurIPS (2016) 2, 5
53. Vu, T.H., Osokin, A., Laptev, I.: Context-aware cnns for person head detection. In: ICCV (2015) 20
54. Wang, B., Zhang, L., Wen, L., Liu, X., Wu, Y.: Towards real-world prohibited item detection: A large-scale x-ray benchmark. In: CVPR (2021) 21
55. Wang, J., Min, W., Hou, S., Ma, S., Zheng, Y., Jiang, S.: LogoDet-3K: A large-scale image dataset for logo detection. arXiv preprint arXiv:2008.05359 (2020) 6, 20
56. Wang, X., Huang, T.E., Darrell, T., Gonzalez, J.E., Yu, F.: Frustratingly simple few-shot object detection. In: ICML (2020) 2, 3, 5, 8, 9, 10, 11, 12, 14, 19, 22, 23, 24, 26
57. Wang, X., Cai, Z., Gao, D., Vasconcelos, N.: Towards universal object detection by domain attention. In: CVPR (2019) 5, 21
58. Wang, Y.X., Ramanan, D., Hebert, M.: Meta-learning to detect rare objects. In: ICCV (2019) 3, 4
59. Wu, X., Sahoo, D., Hoi, S.: Meta-rcnn: Meta learning for few-shot object detection. In: Proceedings of the 28th ACM International Conference on Multimedia (2020) 3
60. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019) 9
61. Xia, G.S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., Datcu, M., Pelillo, M., Zhang, L.: DOTA: A large-scale dataset for object detection in aerial images. In: CVPR (2018) 20

62. Yan, K., Wang, X., Lu, L., Zhang, L., Harrison, A.P., Bagheri, M., Summers, R.M.: Deep lesion graphs in the wild: relationship learning and organization of significant radiology image findings in a diverse large-scale lesion database. In: CVPR (2018) 21
63. Yan, X., Chen, Z., Xu, A., Wang, X., Liang, X., Lin, L.: Meta R-CNN: Towards general solver for instance-level low-shot learning. In: ICCV (2019) 4
64. Yang, S., Luo, P., Loy, C.C., Tang, X.: Wider face: A face detection benchmark. In: CVPR (2016) 6, 20
65. Yousif, H., Kays, R., He, Z.: Dynamic programming selection of object proposals for sequence-level animal species classification in the wild. IEEE Transactions on Circuits and Systems for Video Technology (2019) 20
66. Zhang, L., Zhou, S., Guan, J., Zhang, J.: Accurate few-shot object detection with support-query mutual guidance and hybrid loss. In: CVPR (2021) 5
67. Zhang, S., Xie, Y., Wan, J., Xia, H., Li, S.Z., Guo, G.: WiderPerson: A diverse dataset for dense pedestrian detection in the wild. IEEE Transactions on Multimedia **22**(2), 380–393 (2019) 6, 21
68. Zhou, X., Girdhar, R., Joulin, A., Krähenbühl, P., Misra, I.: Detecting twenty-thousand classes using image-level supervision. arXiv preprint arXiv:2201.02605 (2021) 4, 9, 14
69. Zhou, X., Koltun, V., Krähenbühl, P.: Probabilistic two-stage detection. arXiv preprint arXiv:2103.07461 (2021) 3, 8, 9, 13
70. Zhou, X., Koltun, V., Krähenbühl, P.: Simple multi-dataset detection. In: CVPR (2022) 4, 8, 9, 13
71. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019) 3, 9, 13
72. Zhu, P., Wen, L., Du, D., Bian, X., Fan, H., Hu, Q., Ling, H.: Detection and tracking meet drones challenge. IEEE TPAMI (2021). https://doi.org/10.1109/TPAMI.2021.3119563 6, 20
73. Zhu, X., Hu, H., Lin, S., Dai, J.: Deformable convnets V2: more deformable, better results. In: CVPR (2019) 8, 9, 21
74. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: Deformable transformers for end-to-end object detection. In: ICLR (2021) 3, 8, 9, 13
75. Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., Hu, S.: Traffic-sign detection and classification in the wild. In: CVPR (2016) 21
76. Ziller, A., Hansjakob, J., Rusinov, V., Zügner, D., Vogel, P., Günnemann, S.: Oktoberfest food dataset. arXiv preprint arXiv:1912.05007 (2019) 6, 20

A Natural *K*-Shot Sampling

In this section, we describe how we perform natural K-shot sampling in detail:

Step 1. Sample $C \times K$ images. C is the number of classes of the original dataset $\mathcal{S}$. In this step, without worrying about class labels, we sample $\mathcal{S}$ from the entire dataset $\mathcal{D}$. Unlike the standard K-shot sampling algorithm in recent FSOD works [26,56,47], we do not apply stratified sampling. This is because an image usually contains multiple annotations, such that stratified sampling might result in an artificial class distribution [26].

Step 2. Check missing classes. The initial sampled dataset might not contain some classes, particularly those present only in a few images in the original dataset. To compensate for this, we check if there are any missing classes and update the sampled dataset. Specifically, we manage two datasets: $\mathcal{P}$ is a set of images to be added, and $\mathcal{Q}$ is a set of images to be kept. Then, for each class, if no image in $\mathcal{S}$ contains the class, we sample an image from the $\mathcal{D}$ containing the class and put it in $\mathcal{P}$; otherwise, we sample an image from $\mathcal{S}$ containing the class and put it in $\mathcal{Q}$.

Step 3. Update the sampled dataset. As the final step, we adjust the initial dataset $\mathcal{S}$ to guarantee that all classes are present. To match the number of added and removed images, we sample a set of images to be removed $\mathcal{R}$ from $\mathcal{S} - \mathcal{Q}$ where the size of $\mathcal{R}$ is the same as $\mathcal{P}$. Here, $\mathcal{Q}$ guarantees that any class in $\mathcal{S}$ does not become empty. Finally, we add $\mathcal{P}$ and remove $\mathcal{R}$ from $\mathcal{S}$.

The complete algorithm is in Algorithm A.1.

B Dataset Size Reduction

We initially collected more than 100 public detection datasets, and then selected 32 datasets based on availability, diversity of domains, annotation quality, and number of citations. After initial experiments on them, to reduce the computational burden for future research, we picked 10 datasets out of the 32, which show similar performance trends with the 32 datasets, while covering a variety of domains based on the domain distance.

In the proposed MoFSOD benchmark, several datasets contain a large number of classes and testing images, such as LogoDet-3K. With the proposed natural K-shot sampling, the training time is proportional to the number of classes. To address concerns on computational cost and speed up overall experiment time, we limited the number of classes to 50 and the number of test samples to 1k.

Specifically, we randomly sample 50 classes and remove images containing all the rest classes in each episode, such that the intention of the original datasets is kept, *i.e.*, all remaining logos or traffic signs should be detected. We note that all classes in these datasets are mostly isolated to certain images, such that removing images containing a class does not hurt the distribution of other classes. We confirmed that the performance differences between sampled and full test sets are less than 1.5% for all datasets.

Algorithm A.1 Natural K-shot sampling algorithm.

Input: Dataset $\mathcal{D}$, classes $\mathcal{Y}$, number of classes $C = |\mathcal{Y}|$, average shot number K
Output: Sampled dataset $\mathcal{S}$
if $|\mathcal{D}| \leq C \times K$ **then**
  $\mathcal{S} \leftarrow \mathcal{D}$
else
  // Step 1: sample an initial dataset
  Sample $\mathcal{S} \subset \mathcal{D}$ where $|\mathcal{S}| = N \times K$
  // Step 2: check missing classes
  $\mathcal{P} = \{\}$ // images to be added
  $\mathcal{Q} = \{\}$ // images to be kept
  for $y \in \mathcal{Y}$ **do**
    if no image in $\mathcal{S}$ contains y **then**
      Sample $I \in \mathcal{D}$ where I contains y
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{I\}$
    else
      Sample $I \in \mathcal{S}$ where I contains y
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{I\}$
    end if
  end for
  // Step 3: update the sampled dataset
  if $|\mathcal{P}| > 0$ **then**
    Sample $\mathcal{R} \subset \mathcal{S} - \mathcal{Q}$ where $|\mathcal{R}| = |\mathcal{P}|$
    $\mathcal{S} \leftarrow (\mathcal{S} \cup \mathcal{P}) - \mathcal{R}$
  end if
end if

C Additional Experimental Results

C.1 Dataset Statistics

More detailed statistics of the ten datasets of MoFSOD can be found in Table C.1.

C.2 Detailed 1-, 3- and 10-shot Results

In addition to per-dataset 5-shot results in Table 3 of the main paper, we present per-dataset 1-, 3- and 10-shot results in Table C.2, C.3, and C.4, respectively.

C.3 Extended 32 Datasets Results

We evaluate *FT* against SOTA methods in an extended 32 dataset benchmark on 17 Domains. These datasets are: CARPK [22], DOTA [61], and VisDrone [72] in aerial images, DeepFruits [41] and MinneApple [19] in agriculture, ENA24 [65] and iWildCam [1] in animal in the wild, Clipart, Comic, and Watercolor [24] in cartoon, SKU110K [14] in dense product, DeepFashion2 [11] and iMaterialist [16] in fashion, WIDER FACE [64] in face, Kitchen [13] and Oktoberfest [76] in food, HollywoodHeads [53] in head, LogoDet-3K [55] and OpenLogo [46] in logo,

Table C.1: Statistics of 10 datasets in the proposed benchmark. For KITTI We use the merged set of classes from the universal object detection benchmark [57].

Domain	Dataset	# classes	# train images	# train anno.	# test images	# test anno.
Aerial	VisDrone	10	7019	381965	1610	75103
Agriculture	DeepFruits	7	457	2553	114	590
Animal	iWildCam	1	21065	31591	5313	7901
Cartoon	Clipart	20	500	1640	500	1527
Fashion	iMaterialist	46	45623	333402	1158	8782
Food	Oktoberfest	15	1110	2697	85	236
Logo	LogoDet-3K	352	18752	35264	8331	15945
Person	CrowdHuman	2	15000	705967	4370	206231
Security	SIXray	5	7496	15439	1310	2054
Traffic	KITTI	4	5481	38077	7481	52458

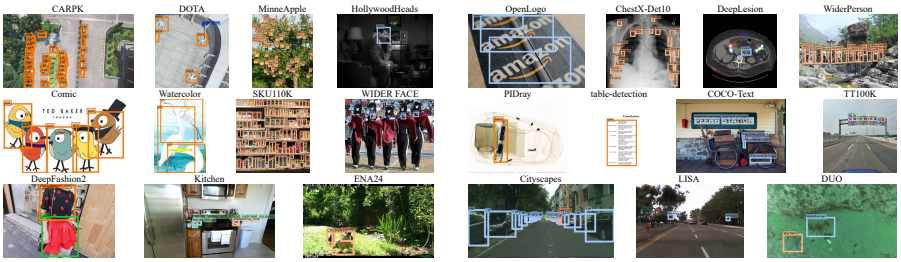


Fig. C.1: Image samples of the additional datasets.

ChestX-Det10 [33] and DeepLesion [62] in medical imaging, CrowdHuman [44] and WiderPerson [67] in person, PIDray [54] and SIXray [34] in security, table-detection [42] in table, COCO-Text [51] in text in the wild, and Cityscapes [5], KITTI [12], LISA [35], and TT100K [75] in traffic, DUO [32] in underwater. Their statistics can be found in Table C.5. Sample images from these datasets can be found in Figure C.1.

Table C.6 compares *FT* and SOTA methods. *TFA-cos* is a variation of *TFA* where the classification head is replaced with the cosine similarity. Note that, while the comparison within tables is fair, the results are NOT directly comparable to the results in the main paper, as they are experimented in different settings. Specifically, the architecture uses deformable convolution v1 while the one in the main paper uses v2, and is trained with a non-standard scheduler. We can observe that *FT* is a strong baseline, outperforming all other methods.

We then present the results between different architectures and pre-training datasets in Table C.7. Although the ablation study here is not as comprehensive as the main paper, we can still see that *Cascade R-CNN-P67* outperforms *Faster R-CNN*. The margin here is smaller mainly due to the less optimal learning rate scheduler we used for *Cascade R-CNN-P67* and possibly the lack of deformable convolution in this model. Once we use the same learning rate scheduler and backbone architecture with deformable convolution v2 [73] for both *Cascade R-CNN-P67* and *Faster R-CNN* as in the main paper, the performance gap for different shots actually increases. On the other hand, the comparison between all *Cascade R-CNN-P67* experiments is fair. We can see that over a larger range of

Table C.2: Per-dataset 1-shot performance of the effects of tuning different parameters, different architectures and pre-training datasets.

1-shot	Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Unfrozen	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Last FC layers (TFA [56])	7.5 ± 0.5	28.7 ± 5.0	55.5 ± 17.3	29.2 ± 2.5	6.7 ± 1.2	21.9 ± 3.3	12.3 ± 3.9	26.3 ± 2.1	2.6 ± 1.8	43.6 ± 5.7	23.4 ± 4.6	2.7 ± 0.3
Detection Head (FSCE-base [47])	7.8 ± 0.8	34.6 ± 6.3	62.3 ± 8.2	30.6 ± 2.6	14.1 ± 1.2	41.0 ± 4.0	24.1 ± 5.5	44.4 ± 4.8	4.5 ± 2.4	41.3 ± 5.5	30.5 ± 2.3	1.9 ± 0.1
Whole Network (Ours-FT)	8.4 ± 1.0	36.2 ± 7.7	56.1 ± 5.1	37.2 ± 3.9	14.2 ± 1.5	47.7 ± 6.7	25.0 ± 4.7	45.0 ± 4.2	6.6 ± 4.3	38.8 ± 3.6	**31.5** ± 2.0	**1.5** ± 0.3

(a) Fine-tuning different number of parameters with Faster R-CNN pre-trained on COCO.

1-shot		Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Architecture	Pre-training	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Faster R-CNN	COCO	8.4 ± 1.0	36.2 ± 7.7	56.1 ± 5.1	37.2 ± 3.9	14.2 ± 1.5	47.7 ± 6.7	25.0 ± 4.7	45.0 ± 4.2	6.6 ± 4.3	38.8 ± 3.6	31.5 ± 2.0	3.6 ± 0.2
Cascade R-CNN		7.2 ± 0.8	35.2 ± 6.4	56.3 ± 8.0	39.0 ± 3.6	13.0 ± 1.2	47.1 ± 6.4	27.6 ± 4.3	44.5 ± 4.2	6.5 ± 4.5	38.3 ± 5.6	31.5 ± 2.1	3.9 ± 0.4
CenterNet2		7.9 ± 0.8	35.6 ± 5.2	38.2 ± 20.6	33.0 ± 7.7	14.7 ± 2.2	45.2 ± 6.7	27.6 ± 4.3	43.8 ± 4.8	7.0 ± 4.1	38.4 ± 3.6	29.1 ± 5.2	4.0 ± 0.3
RetinaNet		5.5 ± 0.6	27.6 ± 7.3	50.9 ± 14.6	10.2 ± 1.7	8.7 ± 0.7	42.6 ± 5.8	24.3 ± 4.1	41.8 ± 2.8	6.3 ± 3.9	36.3 ± 2.3	25.4 ± 4.0	5.4 ± 0.6
Deformable-DETR		8.7 ± 1.1	44.0 ± 6.4	53.2 ± 12.1	23.4 ± 4.4	15.3 ± 1.3	47.8 ± 6.3	28.0 ± 4.9	47.8 ± 4.4	10.1 ± 4.6	41.9 ± 5.1	32.0 ± 2.9	2.5 ± 0.5
Cascade R-CNN-P67		9.6 ± 1.0	41.0 ± 4.9	65.5 ± 7.2	44.0 ± 2.5	16.2 ± 1.6	51.3 ± 5.9	28.3 ± 4.9	47.0 ± 3.9	8.8 ± 4.5	42.1 ± 4.5	**35.4** ± 1.8	**1.6** ± 0.3
Faster R-CNN	LVIS	8.3 ± 0.7	45.2 ± 4.4	58.7 ± 6.4	24.2 ± 3.9	20.7 ± 1.3	49.2 ± 5.2	25.6 ± 5.2	41.6 ± 3.2	8.6 ± 3.8	34.9 ± 3.5	31.7 ± 1.7	2.1 ± 0.2
CenterNet2		7.6 ± 0.7	41.5 ± 6.0	36.0 ± 12.7	20.0 ± 2.4	18.7 ± 1.4	50.4 ± 7.1	27.8 ± 5.0	38.7 ± 2.6	8.6 ± 4.2	32.3 ± 3.6	28.1 ± 3.3	2.7 ± 0.3
Cascade R-CNN-P67		9.2 ± 0.8	46.4 ± 6.1	61.2 ± 6.0	29.9 ± 2.9	23.1 ± 1.4	52.4 ± 6.9	31.0 ± 5.3	43.4 ± 2.7	9.2 ± 4.7	38.5 ± 4.8	**34.4** ± 2.0	**1.2** ± 0.2

(b) Performance of different architectures pre-trained on COCO and LVIS.

1-shot		Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Architecture	Pre-training	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Cascade R-CNN-P67	ImageNet	4.8 ± 0.5	23.5 ± 4.5	1.0 ± 0.8	2.8 ± 1.0	9.3 ± 1.1	43.3 ± 5.2	23.1 ± 3.8	14.8 ± 2.5	2.1 ± 1.3	10.7 ± 2.7	13.5 ± 1.6	6.0 ± 0.1
	COCO	9.6 ± 1.0	41.0 ± 4.9	65.5 ± 7.2	44.0 ± 2.5	16.2 ± 1.6	51.3 ± 5.9	28.3 ± 4.9	47.0 ± 3.9	8.8 ± 4.5	42.1 ± 4.5	**35.4** ± 1.8	2.9 ± 0.4
	FSODD	5.8 ± 0.5	48.0 ± 3.9	44.1 ± 11.1	12.5 ± 2.7	14.8 ± 1.2	50.1 ± 6.5	29.3 ± 4.2	28.6 ± 1.8	8.0 ± 3.6	25.5 ± 3.0	26.7 ± 2.9	4.2 ± 0.5
	LVIS	9.2 ± 0.8	46.4 ± 6.1	61.2 ± 6.0	29.9 ± 2.9	23.1 ± 1.4	52.4 ± 6.9	31.0 ± 5.3	43.4 ± 2.7	9.2 ± 4.7	38.5 ± 4.8	34.4 ± 2.0	3.0 ± 0.2
	Unified	9.7 ± 1.0	47.2 ± 6.5	45.8 ± 8.5	31.2 ± 5.1	18.4 ± 1.3	52.8 ± 6.1	31.2 ± 5.1	46.4 ± 3.0	10.4 ± 4.8	39.5 ± 3.6	33.3 ± 2.2	2.8 ± 0.3
	LVIS+	11.7 ± 1.0	57.4 ± 6.4	30.9 ± 15.9	37.3 ± 1.9	25.5 ± 0.9	50.0 ± 8.0	36.0 ± 3.8	45.1 ± 3.1	13.3 ± 5.2	39.5 ± 4.9	34.7 ± 4.2	**2.1** ± 0.5
CenterNet2	COCO	7.9 ± 0.8	35.6 ± 5.2	38.2 ± 20.6	33.0 ± 7.7	14.7 ± 2.2	45.2 ± 6.7	27.6 ± 4.3	43.8 ± 4.8	7.0 ± 4.1	38.4 ± 3.6	29.1 ± 5.2	3.1 ± 0.4
	LVIS	7.6 ± 0.7	41.5 ± 6.0	36.0 ± 12.7	20.0 ± 2.4	18.7 ± 1.4	50.4 ± 7.1	27.8 ± 5.0	38.7 ± 2.6	8.6 ± 4.2	32.3 ± 3.6	28.1 ± 3.3	3.3 ± 0.3
	LVIS+	10.6 ± 1.1	55.7 ± 5.4	38.4 ± 12.8	35.5 ± 2.9	25.1 ± 1.1	48.4 ± 6.7	36.7 ± 4.2	46.4 ± 3.3	13.9 ± 5.7	37.9 ± 5.3	34.9 ± 3.2	1.9 ± 0.3
	LVIS++	10.7 ± 1.0	59.4 ± 5.5	41.7 ± 13.4	38.2 ± 2.3	26.7 ± 1.0	46.2 ± 6.1	35.6 ± 4.2	47.0 ± 3.5	15.7 ± 5.8	37.1 ± 4.7	**35.8** ± 3.4	**1.7** ± 0.3

(c) Performance of Cascade R-CNN-P67 and CenterNet2 pre-trained on different datasets.

domains, **Unified** provides better results than **COCO** by a significant margin. However, this performance gap could be due to the non-optimal training of **COCO**. These suggest that besides the size/quality of the pre-training datasets, how to train for downstream tasks optimally is also an important factor.

Table C.3: Per-dataset 3-shot performance of the effects of tuning different parameters, different architectures and pre-training datasets.

3-shot	Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Unfrozen	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Last FC layers (TFA [56])	9.4 ± 0.4	41.8 ± 3.0	70.3 ± 4.5	35.9 ± 2.3	7.7 ± 1.3	32.0 ± 6.9	13.3 ± 3.3	29.6 ± 1.0	5.4 ± 1.9	46.6 ± 4.2	29.2 ± 1.8	2.7 ± 0.2
Detection Head (FSCE-base [47])	11.4 ± 0.7	51.6 ± 4.8	70.0 ± 1.9	38.7 ± 1.7	18.4 ± 1.0	61.4 ± 5.4	38.4 ± 4.8	49.0 ± 3.2	10.7 ± 3.5	44.6 ± 4.4	39.4 ± 1.6	1.9 ± 0.2
Whole Network (Ours-FT)	12.0 ± 0.8	52.6 ± 4.6	62.9 ± 5.2	45.5 ± 3.1	19.3 ± 1.0	70.1 ± 5.9	41.7 ± 4.5	48.8 ± 2.7	15.5 ± 6.2	43.2 ± 3.6	**41.1** ± 1.8	**1.5** ± 0.2

(a) Fine-tuning different number of parameters with Faster R-CNN pre-trained on COCO.

3-shot		Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Architecture	Pre-training	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Faster R-CNN	COCO	12.0 ± 0.8	52.6 ± 4.6	62.9 ± 5.2	45.5 ± 3.1	19.3 ± 1.0	70.1 ± 5.9	41.7 ± 4.5	48.8 ± 2.7	15.5 ± 6.2	43.2 ± 3.6	41.1 ± 1.8	3.5 ± 0.3
Cascade R-CNN	COCO	11.0 ± 0.7	52.3 ± 2.9	67.7 ± 3.2	45.9 ± 2.5	18.3 ± 0.9	69.5 ± 5.1	42.3 ± 4.1	48.9 ± 2.6	14.4 ± 5.9	41.8 ± 3.1	41.2 ± 1.5	3.8 ± 0.3
CenterNet2	COCO	11.6 ± 0.7	50.9 ± 6.3	60.6 ± 4.9	44.0 ± 6.8	19.7 ± 2.4	68.2 ± 5.9	42.7 ± 2.4	48.6 ± 3.9	15.5 ± 5.7	40.5 ± 4.4	40.2 ± 1.9	3.8 ± 0.4
RetinaNet	COCO	8.2 ± 0.5	45.7 ± 3.2	59.0 ± 7.2	19.2 ± 1.8	14.5 ± 0.6	66.5 ± 6.4	39.1 ± 4.7	45.5 ± 2.0	12.0 ± 4.9	37.8 ± 2.9	34.8 ± 2.2	5.6 ± 0.6
Deformable-DETR	COCO	12.7 ± 0.7	61.1 ± 4.3	64.8 ± 3.2	35.9 ± 2.7	19.9 ± 1.2	67.9 ± 4.6	42.5 ± 4.6	53.1 ± 3.1	21.3 ± 6.7	43.4 ± 3.5	42.3 ± 1.6	2.7 ± 0.5
Cascade R-CNN-P67	COCO	13.7 ± 0.9	55.3 ± 3.0	72.8 ± 2.5	52.1 ± 2.4	21.9 ± 1.0	71.2 ± 5.4	46.6 ± 4.4	51.2 ± 2.4	17.9 ± 5.6	44.4 ± 2.8	**44.7** ± 1.6	**1.7** ± 0.5
Faster R-CNN	LVIS	11.9 ± 0.6	59.2 ± 5.2	69.4 ± 3.1	33.8 ± 3.5	25.8 ± 0.9	71.1 ± 4.4	41.5 ± 4.2	45.9 ± 2.5	18.3 ± 5.1	38.6 ± 3.6	41.6 ± 1.5	2.1 ± 0.3
CenterNet2	LVIS	11.2 ± 0.6	56.9 ± 3.9	59.0 ± 5.5	27.9 ± 3.9	23.1 ± 0.6	70.7 ± 5.0	45.3 ± 4.0	43.8 ± 2.3	16.5 ± 5.3	35.8 ± 4.2	39.0 ± 1.7	2.7 ± 0.3
Cascade R-CNN-P67	LVIS	13.1 ± 0.7	59.9 ± 5.3	71.1 ± 3.1	39.6 ± 3.7	28.1 ± 1.0	71.9 ± 5.5	47.7 ± 2.8	47.3 ± 2.4	19.0 ± 5.4	42.8 ± 4.1	**44.0** ± 1.6	**1.2** ± 0.3

(b) Performance of different architectures pre-trained on COCO and LVIS.

3-shot		Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Architecture	Pre-training	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Cascade R-CNN-P67	ImageNet	7.9 ± 0.5	43.1 ± 3.3	4.1 ± 2.2	7.3 ± 1.8	16.7 ± 0.7	65.8 ± 5.5	37.6 ± 4.7	25.6 ± 3.3	6.5 ± 3.4	17.9 ± 2.0	23.2 ± 1.5	6.0 ± 0.1
	COCO	13.7 ± 0.9	55.3 ± 3.0	72.8 ± 2.5	52.1 ± 2.4	21.9 ± 1.0	71.2 ± 5.4	46.6 ± 4.4	51.2 ± 2.4	17.9 ± 5.6	44.4 ± 2.8	44.7 ± 1.6	2.9 ± 0.4
	FSODD	9.0 ± 0.8	61.0 ± 3.4	62.3 ± 4.6	19.6 ± 2.5	19.5 ± 0.8	71.8 ± 5.0	46.9 ± 4.4	35.0 ± 2.5	16.1 ± 5.0	28.3 ± 3.2	36.9 ± 1.5	4.3 ± 0.4
	LVIS	13.1 ± 0.7	59.9 ± 5.3	71.1 ± 3.1	39.6 ± 3.7	28.1 ± 1.0	71.9 ± 5.5	47.7 ± 2.8	47.3 ± 2.4	19.0 ± 5.4	42.8 ± 4.1	44.0 ± 1.6	3.2 ± 0.5
	Unified	14.0 ± 0.9	62.5 ± 3.2	63.7 ± 4.2	41.9 ± 3.3	24.2 ± 0.8	73.7 ± 5.1	50.1 ± 4.5	50.3 ± 2.2	19.7 ± 4.5	43.1 ± 3.0	44.3 ± 1.4	2.7 ± 0.4
	LVIS+	16.3 ± 0.9	70.7 ± 3.5	55.0 ± 6.3	46.8 ± 2.3	29.8 ± 0.7	72.1 ± 3.5	52.2 ± 3.6	50.1 ± 2.6	26.8 ± 4.9	46.0 ± 3.4	**46.6** ± 1.6	**1.9** ± 0.5
CenterNet2	COCO	7.9 ± 0.8	35.6 ± 5.2	38.2 ± 20.6	33.0 ± 7.7	14.7 ± 2.2	45.2 ± 6.7	27.6 ± 4.3	43.8 ± 4.8	7.0 ± 4.1	38.4 ± 3.6	29.1 ± 5.2	3.1 ± 0.4
	LVIS	7.6 ± 0.7	41.5 ± 6.0	36.0 ± 12.7	20.0 ± 2.4	18.7 ± 1.4	50.4 ± 7.1	27.8 ± 5.0	38.7 ± 2.6	8.6 ± 4.2	32.3 ± 3.6	28.1 ± 3.3	3.3 ± 0.3
	LVIS+	10.6 ± 1.1	55.7 ± 5.4	38.4 ± 12.8	35.5 ± 2.9	25.1 ± 1.1	48.4 ± 6.7	36.7 ± 4.2	46.4 ± 3.3	13.9 ± 5.7	37.9 ± 5.3	34.9 ± 3.2	1.9 ± 0.3
	LVIS++	10.7 ± 1.0	59.4 ± 5.5	41.7 ± 13.4	38.2 ± 2.3	26.7 ± 1.0	46.2 ± 6.1	35.6 ± 4.2	47.0 ± 3.5	15.7 ± 5.8	37.1 ± 4.7	**35.8** ± 3.4	**1.7** ± 0.3

(c) Performance of Cascade R-CNN-P67 and CenterNet2 pre-trained on different datasets.

Table C.4: Per-dataset 10-shot performance of the effects of tuning different parameters, different architectures and pre-training datasets.

10-shot	Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Unfrozen	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Last FC layers (TFA [56])	10.8 ± 0.5	55.9 ± 1.8	73.8 ± 2.9	44.5 ± 1.0	8.1 ± 1.2	48.0 ± 2.7	15.3 ± 4.4	31.4 ± 0.7	11.0 ± 1.2	53.2 ± 2.4	35.2 ± 1.2	2.7 ± 0.2
Detection Head (FSCE-base [47])	15.5 ± 0.6	69.7 ± 2.8	72.2 ± 1.8	50.2 ± 1.1	23.2 ± 1.2	83.0 ± 3.3	55.7 ± 4.3	54.4 ± 1.8	23.8 ± 1.9	54.0 ± 2.5	50.2 ± 1.1	1.8 ± 0.2
Whole Network (Ours-FT)	17.5 ± 0.7	71.4 ± 1.9	65.3 ± 6.9	57.4 ± 1.1	24.8 ± 1.0	90.6 ± 1.9	59.2 ± 4.4	53.3 ± 1.7	36.1 ± 3.0	50.6 ± 3.1	**52.6** ± 1.8	**1.5** ± 0.2

(a) Fine-tuning different number of parameters with Faster R-CNN pre-trained on COCO.

10-shot		Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Architecture	Pre-training	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Faster R-CNN	COCO	17.5 ± 0.7	71.4 ± 1.9	65.3 ± 6.9	57.4 ± 1.1	24.8 ± 1.0	90.6 ± 1.9	59.2 ± 4.4	53.3 ± 1.7	36.1 ± 3.0	50.6 ± 3.1	52.6 ± 1.8	3.8 ± 0.4
Cascade R-CNN	COCO	16.6 ± 0.7	71.6 ± 2.2	69.7 ± 3.5	58.5 ± 1.5	23.8 ± 0.9	90.0 ± 2.2	58.8 ± 4.6	53.5 ± 1.6	33.7 ± 2.7	50.8 ± 2.9	52.7 ± 1.2	3.9 ± 0.4
CenterNet2	COCO	16.4 ± 0.7	71.3 ± 2.0	65.2 ± 5.1	57.6 ± 7.0	25.4 ± 1.7	89.9 ± 2.7	62.0 ± 5.2	54.1 ± 2.8	33.1 ± 2.1	50.0 ± 5.0	52.5 ± 1.9	3.8 ± 0.4
RetinaNet	COCO	12.6 ± 0.6	69.2 ± 2.5	64.3 ± 4.5	40.9 ± 1.6	21.5 ± 0.5	90.2 ± 1.7	60.4 ± 3.5	50.0 ± 1.5	32.7 ± 1.4	46.3 ± 2.4	48.8 ± 1.2	5.2 ± 0.5
Deformable-DETR	COCO	18.3 ± 0.9	78.6 ± 2.0	70.3 ± 3.4	54.5 ± 1.5	24.2 ± 1.1	86.7 ± 2.2	61.1 ± 3.9	59.6 ± 1.7	39.8 ± 3.3	54.3 ± 2.8	54.7 ± 1.0	2.7 ± 0.5
Cascade R-CNN-P67	COCO	19.1 ± 0.8	73.3 ± 1.6	75.4 ± 2.5	63.4 ± 1.0	27.4 ± 1.2	91.2 ± 2.1	65.7 ± 4.7	56.2 ± 1.6	38.3 ± 2.4	53.7 ± 2.9	**56.4** ± 1.1	**1.7** ± 0.4
Faster R-CNN	LVIS	17.7 ± 0.7	74.8 ± 1.9	71.9 ± 2.5	51.1 ± 1.0	31.0 ± 0.8	90.5 ± 1.6	63.1 ± 4.2	51.1 ± 1.7	36.6 ± 2.0	47.7 ± 2.6	53.6 ± 1.0	2.1 ± 0.3
CenterNet2	LVIS	16.6 ± 0.8	74.6 ± 2.0	68.1 ± 3.1	43.9 ± 1.3	28.3 ± 0.9	90.8 ± 2.0	64.1 ± 4.1	50.0 ± 1.6	34.1 ± 1.7	45.7 ± 2.9	51.6 ± 1.0	2.7 ± 0.3
Cascade R-CNN-P67	LVIS	18.6 ± 0.7	76.1 ± 1.4	72.8 ± 2.7	55.7 ± 1.1	33.1 ± 0.6	91.4 ± 2.0	66.9 ± 4.0	52.5 ± 1.5	37.7 ± 2.6	51.1 ± 2.8	**55.6** ± 1.0	**1.2** ± 0.3

(b) Performance of different architectures pre-trained on COCO and LVIS.

10-shot		Aerial	Agriculture	Animal	Cartoon	Fashion	Food	Logo	Person	Security	Traffic	Mean	Rank
Architecture	Pre-training	VisDrone	DeepFruits	iWildCam	Clipart	iMaterialist	Oktoberfest	LogoDet-3K	CrowdHuman	SIXray	KITTI		
Cascade R-CNN-P67	ImageNet	13.5 ± 0.5	66.5 ± 2.5	14.6 ± 4.4	25.8 ± 1.6	21.9 ± 0.6	86.2 ± 3.1	54.7 ± 3.7	38.9 ± 1.3	22.1 ± 2.3	32.6 ± 3.0	37.7 ± 1.2	5.9 ± 0.2
	COCO	19.1 ± 0.8	73.3 ± 1.6	75.4 ± 2.5	63.4 ± 1.0	27.4 ± 1.2	91.2 ± 2.1	65.7 ± 4.7	56.2 ± 1.6	38.3 ± 2.4	53.7 ± 2.9	56.4 ± 1.1	2.9 ± 0.4
	FSODD	13.7 ± 0.6	74.9 ± 2.1	67.8 ± 3.7	35.9 ± 1.9	24.6 ± 0.9	90.8 ± 1.5	66.5 ± 3.6	42.9 ± 1.8	34.3 ± 1.6	39.5 ± 2.6	49.1 ± 1.0	4.5 ± 0.4
	LVIS	18.6 ± 0.7	76.1 ± 1.4	72.8 ± 2.7	55.7 ± 1.1	33.1 ± 0.6	91.4 ± 2.0	66.9 ± 4.0	52.5 ± 1.5	37.7 ± 2.6	51.1 ± 2.8	55.6 ± 1.0	3.2 ± 0.5
	Unified	19.7 ± 0.7	76.8 ± 1.1	69.9 ± 3.4	58.6 ± 1.5	29.5 ± 1.1	92.8 ± 1.0	69.5 ± 4.0	55.6 ± 1.5	39.9 ± 2.9	53.6 ± 2.9	56.6 ± 1.1	2.4 ± 0.2
	LVIS+	21.4 ± 0.7	84.4 ± 1.5	67.1 ± 3.5	60.4 ± 0.8	34.4 ± 0.5	89.9 ± 1.8	70.7 ± 3.2	55.1 ± 1.4	50.2 ± 2.6	55.9 ± 2.9	**59.0** ± 1.0	**2.0** ± 0.4
CenterNet2	COCO	7.9 ± 0.8	35.6 ± 5.2	38.2 ± 20.6	33.0 ± 7.7	14.7 ± 2.2	45.2 ± 6.7	27.6 ± 4.3	43.8 ± 4.8	7.0 ± 4.1	38.4 ± 3.6	29.1 ± 5.2	3.1 ± 0.4
	LVIS	7.6 ± 0.7	41.5 ± 6.0	36.0 ± 12.7	20.0 ± 2.4	18.7 ± 1.4	50.4 ± 7.1	27.8 ± 5.0	38.7 ± 2.6	8.6 ± 4.2	32.3 ± 3.6	28.1 ± 3.3	3.3 ± 0.3
	LVIS+	10.6 ± 1.1	55.7 ± 5.4	38.4 ± 12.8	35.5 ± 2.9	25.1 ± 1.1	48.4 ± 6.7	36.7 ± 4.2	46.4 ± 3.3	13.9 ± 5.7	37.9 ± 5.3	34.9 ± 3.2	1.9 ± 0.3
	LVIS++	10.7 ± 1.0	59.4 ± 5.5	41.7 ± 13.4	38.2 ± 2.3	26.7 ± 1.0	46.2 ± 6.1	35.6 ± 4.2	47.0 ± 3.5	15.7 ± 5.8	37.1 ± 4.7	**35.8** ± 3.4	**1.7** ± 0.3

(c) Performance of Cascade R-CNN-P67 and CenterNet2 pre-trained on different datasets.

Table C.5: Statistics of 32 datasets in the extended benchmark. The 10 datasets used in MoFSOD are shown in **bold**.

Domain	Dataset	# classes	# train images	# train anno.	# test images	# test anno.
Aerial	CARPK	1	989	42275	459	47501
	DOTA	15	8949	116515	8949	116515
	VisDrone	10	7019	381965	1610	75103
Agriculture	**DeepFruits**	7	457	2553	114	590
	MinneApple	1	403	19373	267	8811
Animal	ENA24	22	7031	7811	1758	1963
	iWildCam	1	21065	31591	5313	7901
Cartoon	**Clipart**	20	500	1640	500	1527
	Comic	6	1000	3215	1000	3176
	Watercolor	6	1000	1662	1000	1655
Dense Product	SKU110K	1	8804	1298968	2935	431420
Face	WIDER FACE	1	12880	159423	3222	39698
Fashion	DeepFashion2	13	191961	312187	32153	52491
	iMaterialist	46	45623	333402	1158	8782
Food	Kitchen	11	4711	24730	2016	13430
	Oktoberfest	15	1110	2697	85	236
Head	HollywoodHeads	1	10834	17754	3984	7080
Logo	**LogoDet-3K**	2993	126891	155286	31727	38981
	OpenLogo	352	18752	35264	8331	15945
Medical	ChestX-Det10	10	2320	6864	459	1477
	DeepLesion	1	27289	28871	4831	5122
Person	**CrowdHuman**	2	15000	705967	4370	206231
	WiderPerson	1	8000	245053	1000	28424
Security	PIDray	12	29454	39709	9482	9483
	SIXray	5	7496	15439	1310	2054
Table	table-detection	1	212	244	191	276
Text	COCO-Text	2	19039	163477	4446	37651
Traffic	Cityscapes	8	2965	50348	492	9793
	KITTI	4	5481	38077	7481	52458
	LISA	5	7937	9246	1987	2283
	TT100K	151	6105	16528	3071	8175
Underwater	DUO	4	6617	63999	1100	10518

Table C.6: Performance on the proposed benchmark in AP50 (top) and the average rank (bottom) of different methods on the extended benchmark with 32 datasets. Note that the pre-trained model used for this table is different from the main paper, such that the comparison is fair within these tables only.

Method	1-shot	3-shot	5-shot	10-shot	Mean
TFA [56]	20.6 ± 3.7	25.6 ± 2.0	27.9 ± 1.7	30.9 ± 1.3	26.3 ± 2.4
TFA-cos [56]	20.8 ± 3.6	25.6 ± 2.0	27.7 ± 1.7	30.5 ± 1.3	26.1 ± 2.4
FSCE-base [47]	25.3 ± 4.2	33.6 ± 3.9	37.9 ± 2.0	43.4 ± 1.6	35.1 ± 3.3
FSCE-con [47]	25.8 ± 4.4	34.1 ± 3.7	38.1 ± 1.8	43.3 ± 1.5	35.3 ± 3.3
DeFRCN [36]	25.4 ± 4.0	32.9 ± 3.1	36.7 ± 1.9	41.2 ± 1.9	34.1 ± 3.0
FT	**26.2** ± 3.3	**35.2** ± 3.5	**39.6** ± 2.3	**45.8** ± 2.1	**36.7** ± 2.9

Method	1-shot	3-shot	5-shot	10-shot	Mean
TFA [56]	4.7 ± 0.5	4.8 ± 0.4	4.7 ± 0.3	4.7 ± 0.4	4.7 ± 0.4
TFA-cos [56]	4.3 ± 0.4	4.5 ± 0.4	4.7 ± 0.4	4.8 ± 0.4	4.6 ± 0.4
FSCE-base [47]	3.3 ± 0.4	3.0 ± 0.4	2.9 ± 0.3	2.8 ± 0.3	3.0 ± 0.4
FSCE-con [47]	2.8 ± 0.4	2.7 ± 0.3	2.7 ± 0.3	2.7 ± 0.3	2.7 ± 0.4
DeFRCN [36]	3.3 ± 0.5	3.5 ± 0.5	3.5 ± 0.4	3.7 ± 0.3	3.5 ± 0.5
FT	**2.7** ± 0.4	**2.5** ± 0.4	**2.4** ± 0.5	**2.2** ± 0.5	**2.5** ± 0.5

Table C.7: Performance of FT on the proposed benchmark with different model architectures and pre-training datasets in AP50 (top) and the average rank (bottom) on the 32 datasets extended benchmark. Note that the pre-trained model used for this table is different from the main paper, such that the comparison is fair within these tables only.

Arch	Pre-train	1-shot	3-shot	5-shot	10-shot	Mean
Faster R-CNN	COCO	26.2 ± 3.3	35.2 ± 3.5	39.6 ± 2.3	45.8 ± 2.1	36.7 ± 2.9
Cascade R-CNN-P67	COCO	26.4 ± 3.3	35.7 ± 2.6	40.3 ± 2.0	46.7 ± 1.6	37.3 ± 2.5
	FSODD	23.3 ± 3.2	32.4 ± 2.5	36.8 ± 1.8	42.9 ± 1.7	33.8 ± 2.5
	Unified	**29.2** ± 2.9	**38.7** ± 2.6	**43.4** ± 1.6	**49.7** ± 1.9	**40.3** ± 2.4

Arch	Pre-train	1-shot	3-shot	5-shot	10-shot	Mean
Faster R-CNN	COCO	2.8 ± 0.4	2.9 ± 0.4	3.0 ± 0.3	3.0 ± 0.3	2.9 ± 0.3
Cascade R-CNN-P67	COCO	2.5 ± 0.3	2.5 ± 0.2	2.5 ± 0.3	2.4 ± 0.3	2.5 ± 0.3
	FSODD	3.0 ± 0.3	3.1 ± 0.3	3.3 ± 0.3	3.4 ± 0.3	3.2 ± 0.3
	Unified	**1.6** ± 0.4	**1.4** ± 0.3	**1.3** ± 0.3	**1.2** ± 0.3	**1.4** ± 0.3