Meta-RetinaNet for Few-shot Object Detection

Shaoqi Li[1]
lsqylxq@buaa.edu.cn
Wenfeng Song *[1]
songwenfenga@gmail.com
Shuai Li[1,2]
lishuai@buaa.edu.cn
Aimin Hao[1,2]
ham@buaa.edu.cn
Hong Qin *[3]
qin@cs.stonybrook.edu

[1] State Key Laboratory of Virtual Reality Technology and Systems
Beihang University
Beijing, China

[2] Peng Cheng Laboratory
Shenzhen 518055, China

[3] Department of Computer Science
Stony Brook University
Stony Brook, USA

Abstract

Few shot object detection (FSD) is gaining popularity, enhanced by the deep learning methods in recent years. Meanwhile, meta-learning has achieved great success in few-shot image classification benefitting from its adaptive capability corresponding to a suite of tasks. Yet, most object detection models are based on deep neural networks (DNNs), and they are prone to the overfitting problem due to limited samples available during training. To adapt the learned prior knowledge more effectively to new tasks, this paper proposes a novel Meta-RetinaNet for FSD, which avoids a biased meta-learner and improves its generalization ability. It employs a Meta Coefficient Learner (MCL) trained by the Balanced Loss (BL) to augment the DNNs. Specifically, the MCL adapts to tasks by the product of pre-trained convolution weights and coefficient vectors densely for all the convolutional layers, such that it could adequately transfer the learned knowledge to new tasks (while overcoming the overfitting problem) by training fewer parameters. In addition, the BL expedites the training of a Meta-RetinaNet by balancing the performance of a host of tasks, and it also retains stable performance for new tasks. Our experiments showcase the effectiveness of our method, which achieves the state-of-the-art performance on the multiple settings of Pascal VOC and COCO datasets.

1 Introduction and Motivation

In recent years, deep neural networks (DNNs) have achieved great performance in image classification and object detection [7, 8, 11, 16], however, it generally requires a large amount of data for training. In practice, DNNs tend to overfit when only a few samples are available during training phase. In contrast, rapid learning remains a hallmark of human learning, because human could recognize novel objects from few examples. Inspired by this, few-shot learning is proposed to solve the image classification problem confined by few examples.

* Corresponding Author.

Recently, a line of literatures [5, 13, 15, 18, 23] has documented significant advances in few-shot learning, while only a handful of researches [1, 9, 21, 22] have made progress in the field of few-shot object detection (FSD), where the model determines the localization and category of the objects with a few samples provided. Popular object detection frameworks [11, 16] cannot achieve good performance under FSD due to the limitation of data.

In general, to reduce the dependence of data, most of the meta-learning methods extract features by Shallow Neural Networks (SNNs) in order to avoid overfitting, e.g., MAML [5] using 4-layer convolution. While most detectors are implemented based on DNNs, e.g., Faster RCNN [16] and RetinaNet [11], and overfitting occurs in case of few-shot. Moreover, meta-learning helps the model trained on a variety of tasks being generalizable to new tasks, but it may result in a biased learner which leads to unstable performance in new tasks.

In this paper, we propose a novel Meta-RetinaNet (Fig. 2) based on meta-learning and DNNs for FSD, and its innovations include Meta Coefficient Learner (MCL) and Balanced Loss (BL). Specifically, MCL transfers knowledge without overfitting by reducing trainable parameters and maintaining the complexity of DNNs-based models, where we freeze pre-trained weights and multiply them with dense coefficient vectors which are learnable. Unlike MTL [19], our MCL adds a coefficient vector to a convolutional layer to multiply its weight with learnable bias, while MTL adds two vectors: one for multiplying the weight and one for adding the frozen bias. We simplify the computation process to speed up the network w.r.t. MTL. BL measures the degree of dispersion of the tasks' performance and helps train an unbiased model with more adaptability and stability towards new tasks by minimizing the imbalance. In addition, we use RetinaNet [11] as the baseline which belongs to one-stage detector. The primary contributions of our work are as follows.

- We propose the Meta-RetinaNet for few shot object detection which can be quickly and steadily adapted to new few-shot tasks by rapidly transferring previously-learned detection ability.
- We design a dense MCL mechanism which learns the coefficients of pre-trained layers to optimize new model which efficiently transfers the past to new scenarios. During meta-training, we further propose a novel Balanced Loss that forces tasks to achieve similar performance with an unbiased state and improves the model's stability.
- We conduct extensive experiments on multiple FSD settings of Pascal VOC and COCO, and the ablation study shows the effectiveness of both components (i.e., both MCL and BL).

2 Related Works

Object detection. Thanks to the rise of deep learning, the performance of object detection has been greatly improved in recent years. At present, deep learning-based object detection frameworks are mainly divided into two categories. First, the two-stage detectors can be represented by RCNN series detectors. RCNN [7] uses Convolutional Neural Network (CNN) to extract features of each proposal obtained by selective search and utilizes support vector machines for classification. Fast-RCNN [6] proposes the Region of Interests (RoI) pooling to extract features which improves the performance of RCNN. Faster RCNN [16] introduces the Region Proposal Network (RPN) to generate bounding box proposals. Second, the one-stage frameworks skip proposals generation and predict bounding boxes and categories in

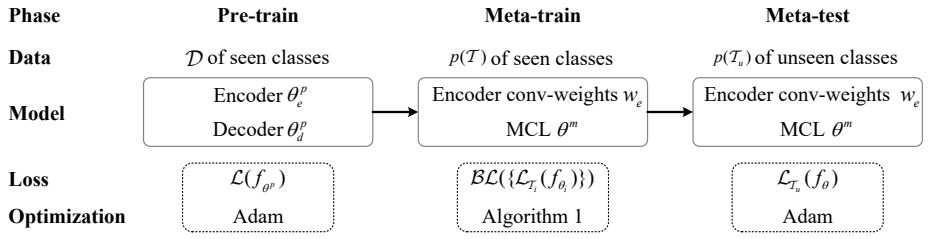

Figure 1: Our pipeline for FSD (see Section 3.1 for a detailed description).

one evaluation. SSD [12] generates anchors with different scales from feature maps and then predicts objects. RetinaNet [11] proposes Focal Loss to solve the problem of extreme foreground-background class imbalance. These detectors are trained by large amounts of data, and are not suitable for solving FSD tasks.

Few-shot learning. Few-shot learning aims to learn new knowledge with only a few examples available. The current mainstream methods are divided into three categories. First, metric learning-based methods [18, 20] learn a distance-based space for classification. Second, model-based methods [14, 17] update model's parameters by its internal architecture or another meta-learner model. Third, gradient-based methods [5, 15, 19] train a meta-learner which can be quickly adapted to new tasks. Specifically, MAML [5] learns an initial model by the second derivative which can be quickly converged to new tasks. MTL [19] designs Scaling and Shifting operations with DNNs being used. Inspired by the few-shot image classification, FSD is proposed to solve the few-shot detection problem which is more challenging.

Few-shot object detection (FSD). There are only a few studies [1, 9, 21, 22] focusing on FSD. LSTD [1] leverages rich base classes knowledge to fine-tune an effective detector on a few examples of unseen tasks with transfer learning. FR [9] proposes a light-weight meta-model by multiplying the last feature map by a number of feature reweighting coefficients, while we multiply every convolutional layer of encoder by a coefficient vector in our method. Meta R-CNN [22] meta-learn over RoI features to adapt tasks. MetaDet [21] trains the category-agnostic model from the base detector and predicts category-specific model by meta-learning. Unlike previous methods, our DNNs-based Meta-RetinaNet contains well-designed architecture and is trained by Balanced Loss works on FSD, which outperforms the baseline and prior methods with greater improvement.

3 Meta-RetinaNet for FSD

We aim to train DNNs based Meta-RetinaNet which could be adapted to new tasks quickly and steadily when doing meta-testing. This section defines the problem and describes the proposed Meta-RetinaNet in details.

3.1 Preliminaries of FSD

We first briefly introduce the problem settings of FSD and relevant notations. Different from the conventional methods, object detection datasets (e.g., Pascal VOC) are split into two parts: seen classes $\mathcal{C}_s$ and unseen classes $\mathcal{C}_u$, where we take advantage of the data of $\mathcal{C}_s$ to train the adaptive model while the data of $\mathcal{C}_u$ are used for evaluation. As shown in Fig. 1, our method for FSD consists of three phases: pre-train, meta-train, and meta-test. Specifically, we get pre-trained feature extractor by training RetinaNet [11] with abundant annotated data

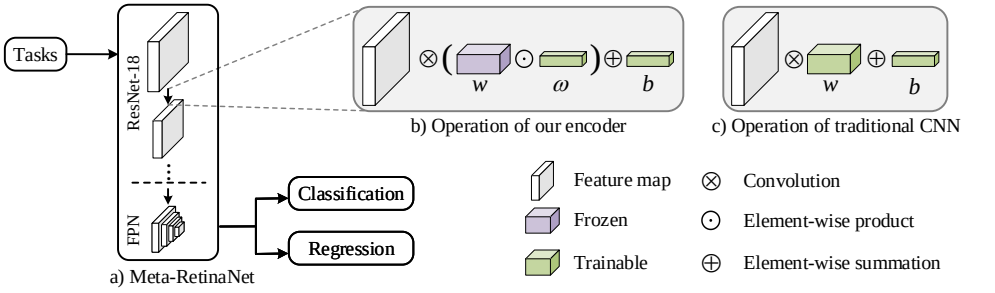

Figure 2: Architecture of Meta-RetinaNet. (a) Meta-RetinaNet is based on RetinaNet and we define a novel feature extraction operation for ResNet-18 & FPN, (b) which requires pre-trained weights w, biases b, and coefficient vector ω initialized to ones, where trainable parameters of Meta-RetinaNet make up the Meta Coefficient Learner (MCL) that is meta-trained by Algorithm 1 with our proposed Balanced Loss (BL). Meta-RetinaNet transfers learned prior knowledge by the operation, (c) while MAML applies 4-layer SNNs as feature extractor with traditional convolution.

of $\mathcal{C}_s$ in the traditional way. We choose RetinaNet because it is an one-stage end-to-end detector which consists of encoder (feature extractor) and decoder (including classification and bounding box regression).

During meta-training, our goal is to meta-train a meta-learner with the data of $\mathcal{C}_s$ which can be quickly converged to new tasks of $\mathcal{C}_u$. We treat the labeled data of $\mathcal{C}_s$ as a distribution $p(\mathcal{T})$, and a task $\mathcal{T}$, or episode, involves the process of meta-updating (lines 3~12 in Algorithm 1). The task can be regarded as a set of samples of randomly-sampled categories. It can be seen that one task $\mathcal{T}$ consists of support set $\mathcal{D}_s$ and query set $\mathcal{D}_q$. For one n-way k-shot task $\mathcal{T}$, where n and k represents the number of object classes and the number of images respectively, there are $n*k$ examples in both of $\mathcal{D}_s$ and $\mathcal{D}_q$. It is worth mentioning that our proposed methods are mainly applied in this phase.

During meta-testing, we aim to evaluate the performance of the meta-trained Meta-RetinaNet on the data of $\mathcal{C}_u$. Similar to meta-train, unseen data is also considered a distribution $p(\mathcal{T}_u)$ and one n-way k-shot task $\mathcal{T}_u$ consists of support set $\mathcal{D}_s^u$ for fine-tuning and query set $\mathcal{D}_q^u$ for evaluation. Then we utilize $\mathcal{D}_s^u$ that contains $n*k$ examples to fine-tune the meta-trained model and $\mathcal{D}_q^u$ is used to evaluate the Meta-RetinaNet's adaptive performance on $\mathcal{C}_u$.

3.2 Meta Coefficient Learner

Most few-shot classification methods based on meta-learning use SNN as feature extractor which contains few learnable parameters in order to avoid overfitting when only a small number of examples are available during training. However, we need to utilize DNNs to get richer features for detection and reduce the size of its trainable parameters simultaneously. This might appear to be contradictory, but it is not true. As shown in Fig. 2, our proposed Meta-RetinaNet recalculates the convolution weights of each layer of the encoder by multiplying the convolution weights of the pre-trained encoder by coefficient vectors. Then we build a new encoder of detector, where each convolutional layer contains a coefficient vector. Therefore, the dense coefficient vectors transfer the previous knowledge for new tasks, and we avoid overfitting with DNNs applied even though only a few examples are available due to fewer trainable parameters. Note that, RetinaNet [11] based on ResNet-18 contains 19.9M

Algorithm 1 Meta-train MCL with BL for FSD.

Require: θ^p: pre-trained parameters of RetinaNet
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: learning rate hyperparameters
1: Initialize θ according to θ^p and Equation 1
2: **while** not done **do**
3: Sample batch of tasks $\{\mathcal{T}_i\} \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K samples $\mathcal{D}_s$ from $\mathcal{T}_i$
6: Evaluate $\mathcal{L}_{\mathcal{T}_i}(f_\theta)$ on $\mathcal{D}_s$
7: Compute adapted parameters with gradient descent: $\theta_i \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample K samples $\mathcal{D}_q$ from $\mathcal{T}_i$
9: Evaluate $\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})$ on $\mathcal{D}_q$
10: **end for**
11: Calculate $\mathcal{BL}(\{\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})\})$ in Equation 5
12: Update $\theta \leftarrow \theta - \beta\nabla_\theta\mathcal{BL}(\{\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})\})$
13: **end while**

trainable parameters, while Meta-RetinaNet only contains $6.9M$ learnable parameters.

We call the trainable substructure (including the dense coefficient vectors, trainable biases of encoder and the whole decoder) Meta Coefficient Learner (MCL). Next, we shall introduce the implementation of the dense MCL in detail. First of all, we pre-train the RetinaNet [10] with ResNet-18 as backbone on data of $\mathcal{C}_s$ (e.g., $|\mathcal{C}_s| = 15$ for Pascal VOC 5-way k-shot) instead of all-classes dataset. For simplicity, we express the model as a function f_{θ^p} which is parameterized by $\theta^p = [\theta^p_e, \theta^p_d]$. We randomly initialize encoder $\theta^p_e = [w^p_e, b^p_e]$ and decoder θ^p_d. Afterwards, we feed data of $\mathcal{C}_s$ to the model and compute its loss (classification & regression) $\mathcal{L}(f_{\theta^p}) = \mathcal{L}_{cls} + \mathcal{L}_{reg}$. Then θ^p is optimized by Adam Optimizer with learning rate decay. In this way, we obtain the pre-trained model θ^p. We further abandon the output layer of classification because the number of categories it outputs may not be same as n-way tasks.

Next, as shown in Fig. 2, we construct our proposed Meta-RetinaNet parameterized by $\theta = [\theta_e, \theta_d]$ on pre-trained model. For i-th convolutional layer in θ_e, its weight and bias are expressed as w_{e_i} and b_{e_i}, respectively. Then we calculate the weight of i-th layer of θ_e according to

$$w_{e_i} = w^p_{e_i} \odot \omega_i, \tag{1}$$

where coefficient vector ω_i is initialized to ones, $\odot$ indicates element-wise product, and pre-trained weight $w^p_{e_i}$ is frozen. We then initialize the bias b_{e_i} of θ_e to $b^p_{e_i}$ and θ_d to θ^p_d, except classification output layer of θ_d which is randomly initialized. Obviously, MCL parameterized by $\theta^m = [\omega, b_e, \theta_d]$ has fewer trainable parameters than the baseline because the shape of ω_i in MCL is $1 \times 1 \times c_i$ while $w^p_{e_i}$ in the baseline is mainly $3 \times 3 \times c_i$, where c_i denotes the number of channels. Thus, Meta-RetinaNet could also be expressed as $\theta = [w^p_e, \theta^m]$, where w^p_e is frozen.

Algorithm 1 explains the complete process of MCL update, and it may be noted that MCL is updated by Algorithm 1 with our proposed Balanced Loss (BL) formulated in Equation 5. We consider a batch of tasks $\{\mathcal{T}_i\} \sim p(\mathcal{T})$. When inner-updating trainable parameters on support set $\mathcal{D}_s$ of task $\mathcal{T}_i$ with gradient descent, Meta-RetinaNet's parameters θ become θ_i, which is formulated as:

$$\theta_i \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}_{\mathcal{T}_i}(f_\theta), \tag{2}$$

where α denotes learning rate of this inner loop. In addition, we can use multiple gradient updates while we only consider once in Algorithm 1 for simplicity. Then we evaluate f_{θ_i} on query set $\mathcal{D}_s$ to get its loss $\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})$. We get a batch of query losses $\{\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})\}$ corresponding to $\{\mathcal{T}_i\}$. More specifically, the objective of meta-training takes the form,

$$\min_{\theta} \mathcal{BL}(\{\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})\}). \tag{3}$$

Next θ is updated across the batch of tasks $\{\mathcal{T}_i\}$ by Stochastic Gradient Descent (SGD),

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \mathcal{BL}(\{\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})\}), \tag{4}$$

where β represents the meta-learning rate. Finally, we repeat the above process until we obtain the most adaptive and stable MCL.

3.3 Balanced Loss

A meta-trained model is expected to be able to quickly adapt to new tasks. Nonetheless, we notice that such adaptability is unstable when only a few samples are available during mate-testing phase. To improve the adaptive stability of the meta-trained model, we propose the Balanced Loss which is used to directly reduce the performance imbalance over tasks during meta-training.

We wish to meta-train the MCL in a balanced way to achieve an unbiased model in which different tasks $\mathcal{T}_i$ in one batch are forced to get close performance over the MCL such that it can be quickly adapted to new tasks with great stability. In other words, we prevent the model from overfitting or underfitting on certain tasks of one batch, which generally is the case caused by previous meta-learning approaches. To achieve that, an unbiased MCL needs to be meta-trained by minimizing the imbalance of tasks' performance. Experimentally, we regard query loss $\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})$ as an indicator of the performance of task $\mathcal{T}_i$ over the MCL. The Balanced Loss (BL) is designed to address the imbalance among tasks,

$$\mathcal{BL}(\{\ell_i\}) = \sum_{i=1}^{\mathcal{M}} (\frac{\ell_i}{\tilde{\ell}})^2, \tag{5}$$

where $\tilde{\ell} = \sum_{i=1}^{\mathcal{M}} \ell_i$ and $\mathcal{M}$ denotes the size (i.e., meta batch size) of set $\{\ell_i\}$.

We can prove that $\mathcal{BL}$ (the value of BL) is minimal iff $\forall i, j \in \{1, 2, ..., \mathcal{M}\}$ and $i \neq j$, s.t. $\ell_i = \ell_j$. Thus, when a task $\mathcal{T}_i$ in one batch is mis-detected, its query loss $\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})$ accounts for a larger proportion to $\mathcal{BL}(\{\ell_i\})$ and then Equation 4 helps $\mathcal{T}_i$ get better performance. In contrast, when $\mathcal{T}_i$ becomes lower $\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})$, i.e., the model fits $\mathcal{T}_i$ well, Equation 4 balances its performance at a moderate level in order to prevent the model from overperforming on $\mathcal{T}_i$. Intuitively, the optimization direction when we apply SGD on $\mathcal{BL}$ is to make the tasks' performance gap smaller, and this is in line with our expectation.

Meta-RetinaNet learns the degree of dispersion of tasks' performance and reduces those performance gaps by meta-updating. Then the meta-trained Meta-RetinaNet could be quickly converged to new tasks with greater stability. It is worth mentioning that our proposed $\mathcal{BL}$ and loss $\mathcal{L}$ of regression & classification are optimized at the same time according to the chain rule.

In general, BL is meant to describe the imbalance of the performance of a batch of tasks during meta-training phase, whose nature is that $\mathcal{BL}$ is minimal iff the input elements (a batch of tasks' detection loss) are equal. The adaptive stability of the meta-trained model

Method	Backbone	S_1					S_2					S_3				
		$k=1$	2	3	5	10	$k=1$	2	3	5	10	$k=1$	2	3	5	10
FR [9]	DarkNet-19	14.8	15.5	26.7	33.9	47.2	15.7	15.3	22.7	30.1	39.2	19.2	21.7	25.7	40.6	41.3
	ResNet-18	14.6	15.6	25.9	33.1	47.5	15.1	15.3	21.5	30.3	38.7	18.8	21.3	25.9	39.5	40.7
Meta R-CNN [23]	ResNet-101	19.9	25.5	35.0	45.7	51.5	10.4	19.4	29.6	34.8	45.4	14.3	18.2	27.5	41.2	48.1
	ResNet-18	11.8	17.4	26.7	33.1	36.9	6.4	11.5	18.1	26.6	29.8	8.2	13.4	19.5	29.8	32.3
MetaDet [23]	VGG-16	18.9	20.6	30.2	36.8	49.6	21.8	23.1	27.8	31.7	43.0	20.6	23.9	29.4	43.9	44.1
RetinaNet-scratch [23]	ResNet-18	0.0	0.0	0.4	0.9	1.0	0.0	0.1	0.6	0.8	1.1	0.0	0.3	0.6	0.9	1.1
RetinaNet-joint [23]		0.1	0.1	0.4	0.5	1.1	0.0	0.1	0.7	0.9	1.3	0.0	0.4	0.9	1.5	2.1
RetinaNet-ft [23]		19.0	27.4	35.2	44.2	53.4	14.1	22.4	26.0	32.0	39.0	21.0	26.7	31.1	38.1	48.0
Meta-RetinaNet w/o BL (ours)	ResNet-18	36.5	48.1	56.2	64.2	70.4	27.0	34.4	40.7	42.5	48.5	35.2	46.1	50.0	56.3	61.2
Meta-RetinaNet (ours)		**38.3**	**51.8**	**59.3**	**65.3**	**71.5**	**28.4**	**36.8**	**42.4**	**45.5**	**50.9**	**35.9**	**48.1**	**53.2**	**58.0**	**63.6**

Table 1: 5-way k-shot FSD performance (mAP%) on the PASCAL VOC dataset with different settings. Meta-RetinaNet consistently outperforms the baseline of three scenarios and other methods with great improvement.

is enhanced by reducing this imbalance with performing SGD on $\mathcal{BL}$ during meta-training, because the BL makes the model trained to a stable state where a batch of tasks get close performance. Then the model could quickly and stably converge to new tasks when you fine-tune the meta-trained model.

4 Experiments, Evaluations, and Discussions

This section evaluates our proposed Meta-RetinaNet with MCL and BL for FSD. First, we describe the settings of used dataset and compare its performance to the baseline [11] of three scenarios and previous methods [9, 21, 22], then we perform ablation studies and analyze its properties.

4.1 Datasets and Settings

We evaluate our method on widely-used object detection datasets, i.e., PASCAL VOC 07 [3] & 12 [4] and COCO [10] datasets.

Intra-domain FSD. PASCAL VOC contains 20 object classes. We meta-train Meta-RetinaNet on PASCAL VOC 07 & 12 train/val set of seen classes and meta-test it on PASCAL VOC 07 test set of unseen classes. Obviously, we need to split 20 object categories into seen classes $\mathcal{C}_s$ and unseen classes $\mathcal{C}_u$. So we select 5 categories as $\mathcal{C}_u$, while $\mathcal{C}_s$ contains the remaining 15 categories. We adopt three different settings of division following the settings in [9, 21, 22]. For n-way k-shot experiments on PASCAL VOC, $n=5$ and $k \in \{1,2,3,5,10\}$. Compared to PASCAL VOC, COCO contains 80 categories and 20 of them are the same as PASCAL VOC. Similarly, we select these 20 categories of COCO as $\mathcal{C}_u$ and the remaining 60 classes to be $\mathcal{C}_s$ and they are non-overlapped, for n-way k-shot experiments on COCO, $n=20$ and $k \in \{10,30\}$. Moreover, we meta-train the model on the $\mathcal{C}_s$ and meta-test it on $\mathcal{C}_u$.

Cross-domain FSD. To evaluate the performance of our method on cross-domain FSD, we perform experiments using both PASCAL VOC and COCO. We use the 20 categories of PASCAL VOC as $\mathcal{C}_u$ and use the remaining 60 classes on COCO as $\mathcal{C}_s$. We focus on 20-way 10-shot tasks, using COCO for training and PASCAL VOC for evaluation.

Baseline. We compare our method with three scenarios based on the original RetinaNet [11]. (1) RetinaNet-scratch: we directly train the n-way k-shot detector on the only $n*k$ examples of $\mathcal{C}_u$ without using the data of $\mathcal{C}_s$. (2) RetinaNet-joint: we jointly train the detector on abundant examples of $\mathcal{C}_s$ and a few examples of $\mathcal{C}_s$. (3) RetinaNet-ft: we pre-train the detector on abundant examples of $\mathcal{C}_s$ and then we fine-tune it to be a n-way k-shot detector with $n*k$ examples of $\mathcal{C}_u$.

#Shots	Method	Average Precision						Average Recall					
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
10	FR [9]	5.6	12.3	4.6	0.9	3.5	10.5	10.1	14.3	14.4	1.5	8.4	28.2
	Meta R-CNN [22]	8.7	19.1	6.6	2.3	7.7	14.0	12.6	17.8	17.9	7.8	15.6	27.2
	MetaDet [21]	7.1	14.6	6.1	1.0	4.1	12.2	11.9	15.1	15.5	1.7	9.7	30.1
	RetinaNet-ft [11]	4.7	8.5	4.5	1.2	4.6	8.5	6.4	9.1	9.4	1.4	8.6	15.7
	Meta-RetinaNet (ours)	**9.7**	**19.9**	**7.7**	**2.8**	**8.5**	**15.2**	**13.4**	**18.9**	**19.0**	**8.3**	**17.1**	**32.8**
30	FR [9]	9.1	19.0	7.6	0.8	4.9	16.8	13.2	17.7	17.8	1.5	10.4	33.5
	Meta R-CNN [22]	12.4	25.3	10.8	2.8	11.6	19.0	15.0	21.4	21.7	8.6	20.0	32.1
	MetaDet [21]	11.3	21.7	8.1	1.1	6.2	17.3	14.5	18.9	19.2	1.8	11.1	34.4
	RetinaNet-ft [11]	9.6	16.4	9.5	2.7	8.9	16.1	10.8	14.8	15.1	3.2	14.2	22.6
	Meta-RetinaNet (ours)	**13.1**	**26.7**	**11.2**	**3.3**	**13.1**	**20.2**	**16.7**	**22.5**	**22.8**	**8.7**	**21.5**	**38.7**

Table 2: 20-way k-shot FSD performance on the COCO dataset. Meta-RetinaNet significantly outperforms the baseline and other approaches.

Implementation Details. Our baseline RetinaNet is built upon ResNet-18 with FPN, and the FSD settings follow those in [9, 21, 22]. We pre-train the baseline on the data of $\mathcal{C}_s$ with Adam optimizer, where the learning rate is 1e-4 (with 0.1 decay), the momentum/momentum2 is 0.9/0.99. We meta-train the MCL with Algorithm 1, where α and β are set to 1e-5, meta batch size $\mathcal{M} = 8$, the number of inner updates in Equation 2 is 1 unless stated otherwise and there are 3000 tasks for VOC and 5000 tasks for COCO. We fine-tune the meta-trained Meta-RetinaNet with $n * k$ examples of n-way k-shot tasks using the same optimizer as pre-training, and then evaluate it with tasks' query set which contains 5 examples for each class. In addition, there are fixed 100 tasks being evaluated rather than one in order to display its effect more stably. All our experiments are performed on Pytorch.

4.2 Few-shot Object Detection

To show the effectiveness of Meta-RetinaNet, we evaluate its adaptability and make comparisons with the baseline and the state-of-the-art approaches on PASCAL VOC and COCO datasets by performing intra-domain FSD and cross-domain FSD.

Pascal VOC. As shown in Table 1, our Meta-RetinaNet based on ResNet-18 outperforms three scenarios based on the original RetinaNet [11] and recent approaches [9, 21, 22] with greater improvement regardless of the split S_i or shot k.

Although RetinaNet achieves good performance on a large amount of labeled images, in the case that only a few examples are available, Meta-RetinaNet shows greater ability to adapt to new tasks compared to three scenarios based on the original RetinaNet. As we can see, based on ResNet-18 pre-trained on ImageNet, RetinaNet trained from scratch works poorly due to overfitting. Even if we use abundant examples of $\mathcal{C}_s$ and a few examples of $\mathcal{C}_s$ to jointly train the RetinaNet, its performance is still not good. RetinaNet-ft performs better than both of the above because of the weak knowledge transfer from fine-tuning.

We pre-train both RetinaNet-ft and Meta-RetinaNet on abundant examples of $\mathcal{C}_s$ and the difference between them is that RetinaNet-ft fine-tunes all the parameters ($19.9M$) while Meta-RetinaNet just meta-trains the MCL ($6.9M$) after freezing the weights of RetinaNet encoder. In other words, our method only needs to learn fewer parameters, reducing the probability of overfitting while obtaining strong features for detection.

Meta-RetinaNet consistently outperforms the baseline [11] and other methods [9, 21, 22] with greater improvement. For fair comparison, we re-implement the approaches [9, 22] with the same backbone, i.e., ResNet-18. Compared to FR [9] and Meta-RCNN [22] based on the same backbone, Meta-RetinaNet improves the performance with about $10\% \sim 20\%$

$\mathcal{M}$	$k=1$	2	3	5	10
2	36.6	48.5	55.9	64.3	70.3
4	37.5	49.9	57.2	64.8	71.1
8	**38.3**	**51.8**	**59.3**	**65.3**	**71.5**

Table 3: FSD performance (mAP%) with different meta batch size $\mathcal{M}$ of BL.

Method	$k=1$	2	3	5	10
RetinaNet-ft	10.6	10.7	10.8	10.9	10.1
Meta-RetinaNet w/o BL (ours)	10.2	10.6	9.7	10.1	9.6
Meta-RetinaNet (ours)	**8.6**	**8.5**	**8.5**	**8.7**	**7.9**

Table 4: The *stb* of 100 tasks' mAP, where *stb* indicates standard deviation times 100. The smaller the value, the better.

for 5-way k-shot tasks of three dataset splits, where $k = 1,2,3,5,10$. It also outperforms ResNet-101 based Meta R-CNN[22] with greater improvement even though we use a shallower feature extraction network. As visualized in Fig. 4, 1-shot Meta-RetinaNet could predict most objects but it may make some mistakes (e.g., classifying cow into bird) or miss some objects (e.g., blocked bus). Surprisingly, Meta-RetinaNet could get good performance with only 5-shot.

COCO. COCO is more challenging than Pascal VOC, which involves 80 object categories and 20 of them are set to be unseen. Following [9, 21, 22], we use 5k examples from the validation set for evaluation and the remaining train/val images for training. We evaluate 20-way 10-shot and 30-shot tasks on COCO benchmark. Table 2 shows that Meta-RetinaNet significantly outperforms the baseline and other methods. The above illustrates the effectiveness of our approach for intro-domain FSD.

Pascal VOC to COCO. We perform 20-way 10-shot experiments to evaluate Meta-RetinaNet's performance on cross-domain FSD. Meta-RetinaNet achieves 39.1% mAP which outperforms 32.3% of FR [9], 34.0% of MetaDet [21] and 37.4% of Meta R-CNN [22]. Our method alleviates the domain shift issues caused by cross domain. So far, our method is effective on the scenarios of both intra-domain FSD and cross-domain FSD.

4.3 Ablation Studies

We conduct comprehensive ablation studies to evaluate the effectiveness of our proposed components. The experiments are performed on S_1 of Pascal VOC.

MCL. We pre-train both RetinaNet-ft and Meta-RetinaNet (w/o BL) on numbers of examples of $\mathcal{C}_s$. We then fine-tune all parameters of RetinaNet-ft, while we just meta-train MCL of Meta-RetinaNet (w/o BL) with query loss $\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})$ instead of $\mathcal{BL}(\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i}))$ after freezing the pre-trained weights of encoder. As shown in Table 1, Meta-RetinaNet (w/o BL) outperforms RetinaNet-ft thanks to the role of well-designed MCL. Through experiments, we have found Meta-RetinaNet works better when using deeper feature extractor, e.g., ResNet-50 improves about $2\% \sim 5\%$ mAP.

BL. As shown in Table 1, Meta-RetinaNet performs better than Meta-RetinaNet (w/o BL), which means that BL is effective in improving the adaptability of the model by mitigating the imbalance among tasks. The BL introduces a new hyper-parameter, meta batch size $\mathcal{M}$, that controls its scope. The performance is improved as $\mathcal{M}$ increases (refer to Table 3). Due to the limitation of GPU memory, the upper limit of $\mathcal{M}$ is 8, and BL yields an average 2.2 mAP improvement. The mAP improvement could be better if $\mathcal{M}$ is higher.

4.4 Meta-RetinaNet Properties

Meta-RetinaNet is designed to quickly converge to new tasks with greater stability. To analyze the properties of its key design methodologies and modules, we conduct experiments

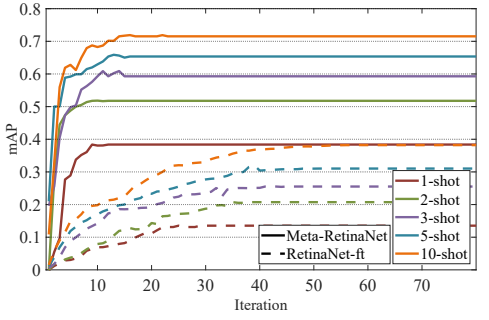

Figure 3: Comparison of adaptation speed (tasks' mAP against the number of iterations) between Meta-RetinaNet and RetinaNet-ft.

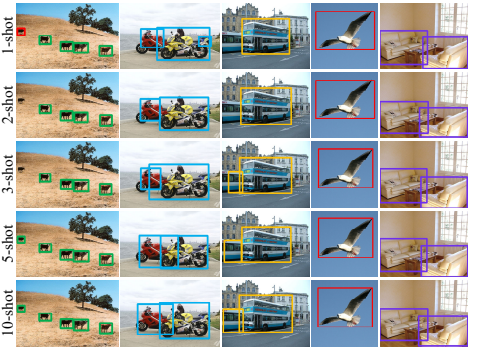

Figure 4: Detection visualization. Colors of detected boxes denote different categories. Meta-RetinaNet could detect objects with only a few training examples.

on S_1 of Pascal VOC.

Fast adaptation. We fine-tune the meta-trained Meta-RetinaNet and the pre-trained RetinaNet with same tasks and learning rate. As shown in Fig. 3, Meta-RetinaNet exhibits fast adaptation ability, where it only requires about 10~15 iterations to fully converge while the baseline requires about 30~50 iterations. The current architecture of Meta-RetinaNet and the meta-training expedite learning from past knowledge and exhibit the ability to be quickly adapted to new tasks. The main reason for this advantage is that, it utilizes meta-learning to start from a more generalized initial model and trains fewer parameters than the baseline.

Stability. To verify the stability of Meta-RetinaNet compared to the baseline RetinaNet-ft, we compute their standard deviation of fixed 100 tasks' mAP on different settings, where we use standard deviation of tasks' performance to indicate the model's stability. As shown in Table 4, Meta-RetinaNet achieves the better stability than the baseline and Meta-RetinaNet (without BL), in which BL plays a vital role that helps train an unbiased model by minimizing BL during meta-training such that the model is more stable when adapting to new tasks.

5 Conclusion

This paper proposes a novel Meta-RetinaNet for FSD with fast adaptability and strong stability. We designed a DNNs based MCL as meta-learner for fast adaptability which transfers past knowledge while avoiding overfitting because of the well-designed architecture. We proposed BL for meta-training so that the meta-trained model could be quickly adapted to new tasks with greater stability. Our newly-added components are verified for their effectiveness, while achieving the best performance compared to the baseline architecture and previous works.

Acknowledgments This research is supported in part by National Key R&D Program of China (No. 2018YFB1700603), National Natural Science Foundation of China (NO. 61672077 and 61532002), Beijing Natural Science Foundation-Haidian Primitive Innovation Joint Fund (L182016), and National Science Foundation of USA (NO. IIS-0949467, IIS-1047715, IIS-1715985, and IIS-1049448).

References

[1] Hao Chen, Yali Wang, Guoyou Wang, and Yu Qiao. Lstd: A low-shot transfer detector for object detection. In *AAAI*, pages 2836–2843, 2018.

[2] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, pages 764–773, 2017.

[3] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

[4] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135, 2017.

[6] Ross Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[9] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *ICCV*, pages 8420–8429, 2019.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014.

[11] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017.

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37, 2016.

[13] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *ICML*, pages 2554–2563, 2017.

[14] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *ICML*, pages 3664–3673, 2018.

[15] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, pages 91–99, 2015.

[17] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, pages 1842–1850, 2016.

[18] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4077–4087, 2017.

[19] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, pages 403–412, 2019.

[20] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, pages 3630–3638, 2016.

[21] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Meta-learning to detect rare objects. In *ICCV*, pages 9925–9934, 2019.

[22] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta r-cnn: Towards general solver for instance-level low-shot learning. In *ICCV*, pages 9577–9586, 2019.

[23] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*.