MDFL: A UNIFIED FRAMEWORK WITH META-DROPOUT FOR FEW-SHOT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Conventional training of deep neural networks usually requires a substantial amount of data with expensive human annotations. In this paper, we utilize the idea of meta-learning to integrate two very different streams of few-shot learning, *i.e.*, the episodic meta-learning-based and pre-train finetune-based few-shot learning, and form a unified meta-learning framework. In order to improve the generalization power of our framework, we propose a simple yet effective strategy named meta-dropout, which is applied to the transferable knowledge generalized from base categories to novel categories. The proposed strategy can effectively prevent neural units from co-adapting excessively in the meta-training stage. Extensive experiments on the few-shot object detection and few-shot image classification datasets, *i.e.*, Pascal VOC, MS COCO, CUB, and mini-ImageNet, validate the effectiveness of our method.

1 INTRODUCTION

Deep Neural Networks (DNNs) have achieved great progress in many computer vision tasks Ren et al. (2016); Dai et al. (2016); Redmon & Farhadi (2017); Lin et al. (2017). However, the impressive performance of these models largely relies on a large amount of data as well as expensive human annotation. When the annotated data are scarce, DNNs cannot generalize well to testing data especially when the testing data belongs to different classes of the training data. In contrast, humans can learn to recognize or detect a novel object quickly with only a few labeled examples. Due to some object categories naturally have few samples or their annotations are extremely hard to obtain, the generalization ability of conventional neural networks is far from satisfactory.

Few-shot learning, therefore, becomes an important research topic to achieve better generalization ability by learning from only a few examples. The mainstream few-shot learning approaches consists of episodic approaches Kang et al. (2019); Yan et al. (2019); Wang et al. (2019); Fan et al. (2020) and pretrain-finetune based approaches Dhillon et al. (2019); Chen et al. (2019); Wang et al. (2020). Episodic meta-learning encapsulates the training samples into an episode Vinyals et al. (2016) to mimic the procedure of few-shot testing. Pre-train finetune-based methods are composed of the pre-training stage and fine-tuning stage, the former is responsible for obtaining a good initialization point from base classes, and the latter adapts the pre-trained model to a specific task, respectively. In order to transfer knowledge from base data to novel ones, both methods are trained with two stages, where the data-sufficient base classes and the data-scarce novel classes are used separately. However, there is no framework to unify the two very different streams, hindering exploring the common and eccentric problem of few-shot learning.

In this paper, we incorporate episodic meta-learning-based which is denoted as episode-based for simplification, and pre-train finetune-based few-shot methods into one unified optimization framework based on the idea of meta-learning. The framework consists of a novel reformulated meta-training stage and a meta-testing stage. In the meta-training stage, our framework considers the common elements of few-shot learning, including meta-knowledge, task-knowledge, meta-loss, task-loss, and the distribution of the data and tasks. In the meta-testing stage, the final model for novel tasks is obtained based on the learned model.

As shown in Figure.1, a deep model can be divided into two components, called task-specific knowledge and transferable knowledge. The former represents the last fully-connected classifier for specific categories, the latter is the well-learned feature representation which is needed to be generalized

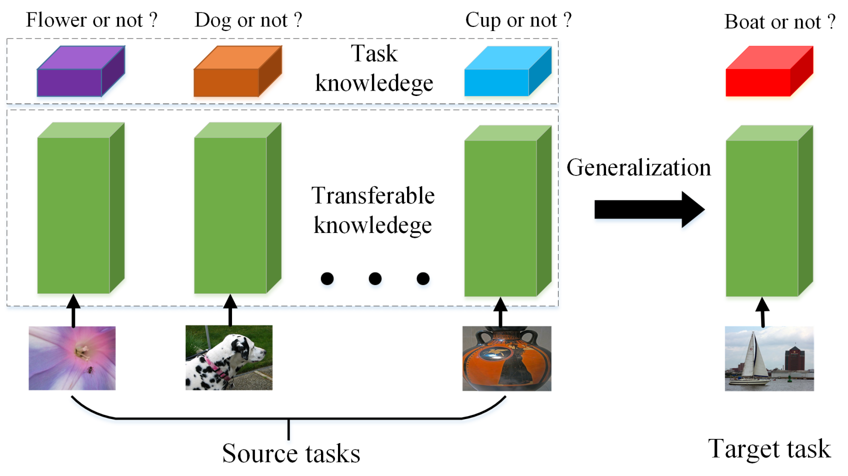


Figure 1: The generalization power of transferable knowledge across different source tasks is the key for few-shot learning, in which transferable knowledge is adapted to the target task.

to novel tasks. Therefore, it is crucial to improve the generalization power of transferable knowledge if we want to apply it from source tasks to novel tasks in few-shot learning. To achieve this goal, we propose a simple yet effective strategy, named meta-dropout. Because our unified framework can integrate two very different streams of few-shot learning and identify which part of a model is the transferable knowledge, the proposed meta-dropout can be easily applied to existing few-shot models. We select several different methods from the above two streams as the baselines to validate the correctness of our framework and the effectiveness of our meta-dropout. By utilizing meta-dropout, our model demonstrates great superiority towards the current excellent methods on few-shot object detection and few-shot image classification tasks. Our contributions can be summarized as three-fold:

- We utilize the idea of meta-learning to integrate two very different streams of few-shot learning, *i.e.*, the episodic meta-learning-based and pre-train finetune-based few-shot learning, and form a unified meta-learning framework.
- We propose a simple yet effective strategy, named meta-dropout, to improve the generalization power of meta-knowledge in our framework.
- Experiments of baselines from different streams evaluate the effectiveness of our approach on the few-shot object detection and image classification datasets, *i.e.*, Pascal VOC, MS COCO, CUB, and mini-ImageNet.

2 Related Work

2.1 Episode-based Few-shot Learning

Few-shot learning is an important yet unsolved task in computer vision Hariharan & Girshick (2017); Koch et al. (2015); Vinyals et al. (2016); Tokmakov et al. (2019). Nowadays, the meta-learning strategy which is called “learning-to-learn” has become an increasingly popular solution. The goal of meta-learning is to obtain task-level meta-knowledge that helps the model quickly generalize across all tasks Andrychowicz et al. (2016); Munkhdalai & Yu (2017); Santoro et al. (2016); Thrun (1998). Recent methods for few-shot learning usually extract meta-knowledge from a set of auxiliary tasks via the episode-based strategy Vinyals et al. (2016), where each episode contains C classes and K samples of each class, *i.e.*, C-way K-shot.

In few-shot image classification, Vinyals et al. (2016) proposed Matching Networks to find the most similar class for the target image among a small set of labeled samples. Prototypical Networks (PN) Snell et al. (2017) extended Matching Networks by producing a linear classifier instead of the weighted nearest neighbor for each class. The cosine similarity-based classifier further enhanced the discriminative power of the trained model Chen et al. (2019); Gidaris & Komodakis (2018). Relation Network (RN) Sung et al. (2018) used a neural network to learn a distance metric, in which the unlabeled images could be classified according to the relation scores between the target sample and a few labeled images. Graph Neural Network (GNN) Kim et al. (2019); Gidaris & Komodakis (2019) was also utilized to model relationships between different categories. In few-shot object detection, Kang et al. (2019) applied a feature re-weighting module to a single-stage object detector (YOLOv2) with the support masks as inputs. Yan et al. (2019) introduced a Predictor-head Remodeling Network (PRN) that shared its backbone with Faster/Mask R-CNN. To disentangle the category-agnostic and category-specific components in a CNN model, Wang et al. (2019) proposed a weight prediction meta-model for predicting the parameters of category-specific components from few samples. Karlinsky et al. (2019) proposed a new module for distance metric learning (DML) that could be used as the classification head combined with the standard object detection model. Fan et al. (2020) introduced a few-shot object detection method, which consisted of an attention-rpn, a multi-relation detector, and a contrastive training strategy.

2.2 Pre-train Finetune-based Few-shot Learning

Pre-train finetune-based approaches are basic yet ignored in few-shot learning due to the excellent performance of episode-based methods. However, some simple pre-train finetune-based methods turn out to be more favorable than many episode-based works Chen et al. (2019); Dhillon et al. (2019); Chen et al. (2021). Chen et al. (2019) introduced a pre-train finetune baseline with a distance-based classifier, achieving a competitive performance with state-of-the-art episode-based classification approaches. Chen et al. (2021) explored a simple process: meta-learning over a whole classification pre-trained model. This simple method achieves competitive performance to state-of-the-art methods on standard benchmarks. By using the proposed regularization, the standard detectors, such as SSD Liu et al. (2016) and FRCNN Ren et al. (2016), were fine-tuned for few-shot problems. Furthermore, Wang et al. (2020) demonstrated that only fine-tuning the last layer of existing models was crucial to the few-shot object detection task. Such a simple approach outperforms the episode-based approaches by 2 to 20 points and even doubles the accuracy of the prior works on current benchmarks.

3 Method

3.1 The Solution of Few-shot problems

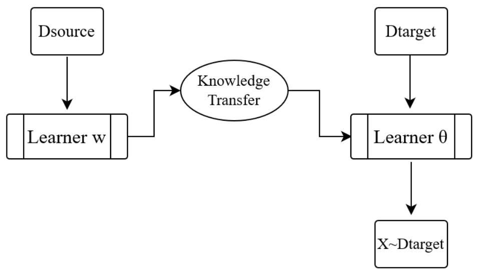
Common supervised learning problems based on abundant training data can be solved by minizing the Equ (1), in which Θ is the trainable parameters of neural networks, and x is input data sampled from $p(D)$. While few-shot tasks are limited in the number of samples, optimizing it directly is likely to results in the overfitting of Θ due to its high-dimensional property.

$$\underset{\Theta}{min} \underset{x \sim p(D)}{E} L(\Theta; x) \tag{1}$$

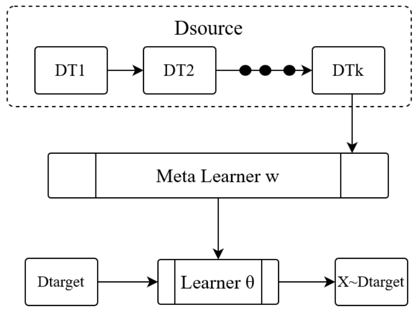
Few-shot learning aims to solve learning problems with just a few training examples. To achieve this goal, the common solution is to reduce the learnable dimension of Θ, and thus the Equ (1) can be re-written as Equ (2), in which Θ = [θ,w]. w represents the useful foundation for few-shot learning, such as a good initailization point or a well-learned feature representation, which is obtained from source tasks. θ are updated on target tasks based on the learned w.

$$\underset{\theta}{min} \underset{x \sim p(D)}{E} L(\theta; x|w) \tag{2}$$

Given a labeled source dataset D_{source}, there are C_{source} source classes with a large number of images in each class. Novel dataset D_{target} with novel classes C_{target} consists of a few samples in each class. C_{source} and C_{target} do not have overlapping categories. The learning goal is adapting the model from D_{source} to D_{target}. D_{source} and D_{target} are used to optimize w and θ respectively. The C-way K-shot few-shot setting that is used for evaluating the performance of a few-shot model means D_{target} has C novel categories and each novel category has K images.



(a) Pre-train finetune-based methods



(b) Episodic meta-learning based methods

Figure 2: The optimization processes of pre-train finetune-based and episodic meta-learning-based methods.

Existing few-shot learning methods consist of two different streams of categories, *i.e.*, the episodic meta-learning-based and pre-train finetune-based methods. However, both streams can be integrated by the above formulation as shown in Figure.2. w in pre-train finetune-based methods, like TFA Wang et al. (2020), is its backbone which is transferred to the second stage for initialization. θ in TFA means the last fully-connected layer that is optimized for novel tasks based on the frozen w. w in Meta R-CNN Yan et al. (2019), which is a representative episodic meta-learning-based method, provides a backbone with good generalization ability. θ in Meta R-CNN is the trainable backbone and the last fully-connected regression and classification layers for novel tasks.

3.2 The Unified Meta-learning Framework

In order to explain the learning process of existing few-shot methods, we propose a unified meta-learning framework to re-formulate Equ (2) as Equ (3), in which the learning goal is to obtain a general transferable knowledge via optimizing the expectation of many source tasks and source datasets. Suppose $p(T)$ presents the distribution of source tasks, in which T consists of infinite basic tasks, and $p(D_t)$ is the distribution of the training dataset D_t of task t, in which the number of the training data is unlimited. w is used to specify meta-knowledge, which is the transferable parameters among different tasks. θ is task-specific knowledge, called task-knowledge. L is the loss function. The overall optimization goal is below:

$$\underset{w}{min} \underset{t\sim p(T)}{E} \underset{x\sim p(D_t)}{E} L(w;\theta_t;x) \tag{3}$$

The goal is to let w get good performance for each task T=$\{D$,$L\}$. Our meta-learning framework consists of a meta-training phase and a meta-testing phase.

3.2.1 Meta-training

The goal of meta-training is formulated below:

$$w* = \arg \underset{w}{max} \log p(w|T) \tag{4}$$

In our meta-training stage, the formulas of optimization are:

$$w* = \arg \underset{w}{min} \underset{t\sim p(T)}{E} \underset{x\sim p(D_t)}{E} L^{meta}(\theta^{*(i)}(w), w) \tag{5}$$

$$s.t.\theta^{*(i)}(w) = \arg \underset{\theta}{min} \underset{D_{ij}\sim p(D_i)}{E} L^{task}(\theta^i, w) \tag{6}$$

By solving the above equations, we can obtain our final optimization formulas below:

$$w* = \arg\min_{w} \int\int L^{meta}(\theta^{*(i)}(w), w)p(T)p(D_t)d_T d_{D_t} \tag{7}$$

$$s.t.\theta^{*(i)}(w) = \arg\min_{\theta} \int L^{task}(\theta^i, w)p(D_t)d_{D_t} \tag{8}$$

w is meta-knowledge and $w*$ is the learned w during the iterations of tasks. θ represents task-knowledge and $\theta*$ is the learned task-knowledge that is used to optimize w. L^{task} is used to optimize the task-specific information and L^{meta} is for generating the best meta-knowledge $w*$. L^{meta} called meta-loss represents which w is good, and L^{task} represents which model is good for a specific task, called task-loss. From formulas (5) and (6), we can obtain the episode-based and pre-train finetune-based frameworks in the following sections.

3.2.2 META-TESTING

Pick a set of Q target tasks to apply the learned w to obtain a task-specific model.

$$D_{target} = \{(D_{target}^{train}, D_{target}^{test})^{(i)}\}_{i=1}^{Q} \tag{9}$$

$$\theta^{*(i)} = \arg\max_{\theta} \log p(\theta|w^*, D_{target}^{train(i)}) \tag{10}$$

D_{target}^{train} and D_{target}^{test} are built from the novel dataset D_{target} with a small scale of classes and samples. D_{target}^{train} corresponds to all annotated data and is used to further optimize the model for the specific tasks. D_{target}^{test} is the real novel test data. $w*$ is the learned best meta-knowledge that is used for obtaining $\theta*$. $\theta*$ is the best task-knowledge for the final prediction.

3.2.3 EPISODE-BASED META-LEARNING FRAMEWORK

Episodic meta-training based methods is optimized in the form of episode, and each episode is corresponding to a specific task. Each episode consists of two parts: support set and query set, which are presented by D_{source}^{train} and D_{source}^{val} in the following formula. D_{source}^{train} and D_{source}^{val} are built from the D_{source}.

$$D_{source} = \{(D_{source}^{train}, D_{source}^{val})^{(i)}\}_{i=1}^{M} \tag{11}$$

Consistent with our unified meta-learning framework, by replacing the dataset D with the D_{source}^{train} and D_{source}^{val}, the formulas of optimization are:

$$w* = \arg\min_{w} \sum_{i=1}^{M}\sum_{j=1}^{N} L^{meta}(\theta^{*(i)}(w), w, D_{source}^{val(ij)}) \tag{12}$$

$$s.t.\theta^{*(i)}(w) = \arg\min_{\theta} \sum_{j=1}^{N} L^{task}(\theta^i, w, D_{source}^{train(ij)}) \tag{13}$$

M is the number of source tasks sampled from $p(T)$ and N means the number of samples in each task. Specifically, when building an episode, some categories of data will be randomly selected to build the D_{source}^{train}, and some samples with the same classes as the D_{source}^{train} will be selected from the remaining data to build the D_{source}^{val}. Specifically, w represents the task-independent network parameters in meta-training stage, and θ is the task-specific network parameters in meta-testing stage. L^{meta} is calculated on the D_{source}^{val} and L^{task} is computed based on D_{source}^{train}.

There are two changes from our unified meta-learning framework to the episode-based formula. First, the number of tasks and the number of samples are limited in each episode, which are presented by M and N respectively. Therefore, using more tasks and more samples by increasing M and N

in meta-training is important for better performance. Second, D^{train}_{source} and D^{val}_{source} are built from a small set of data from the whole dataset, whose distribution is very different from that of the overall dataset. In order to get excellent results, the gap should be reduced. During episode-based training, we can sample data as diverse as possible, to use $[p(x_1), p(x_2), ..., p(x_M)]$ to simulate the real distribution of $p(D_T)$.

3.2.4 Pre-train Finetune-based Meta-learning Framework

Pre-train finetune methods have a two-stage training framework which consists of a base-training stage and a fine-tuning stage. In our meta-learning framework, the base training stage is presented by the meta-training stage and the fine-tuning stage is the meta-testing stage.

In the meta-training stage, a set of M source tasks from $p(T)$ and a set of N samples of each task are sampled to learn w, thereby the formulas of optimization in pre-train finetune-based methods are:

$$w* = \arg\underset{w}{min} \sum_{i=1}^{M} \sum_{j=1}^{N} L^{meta}(\theta^{*(i)}(w), w, D_{source(ij)}) \tag{14}$$

$$s.t.\theta^{*(i)}(w) = \arg\underset{\theta}{min} \sum_{j=1}^{N} L^{task}(\theta^{i}, w, D_{source(ij)}) \tag{15}$$

Consistent with our unified meta-learning framework, the dataset D is replaced with the D_{source}. Pre-train finetune methods assume that pre-training dataset D_{source} is large enough, in order to provide a good initialization for meta-testing. w represents the backbone parameters in the meta-training stage. θ corresponds to the fine-tuning part of the model in the meta-testing phase, which is task-related. The common fine-tuning parts are the final classification and regression layers. The gap between the unified framework and the pre-train finetune-based framework is the scale of the training dataset. Increasing M and N plays a very important role in pre-train finetune-based methods.

3.3 Meta-dropout

Because our unified framework can integrate two different streams of few-shot learning and identify which part of a model is the meta-knowledge, the proposed meta-dropout can be easily applied to many different few-shot models. Meta-dropout means using the idea of dropout Srivastava et al. (2014) on the meta-training stage which is trained with abundant source datasets for improving the generalization power of meta-knowledge. By applying meta-dropout, we can obtain a new formulation of meta-learning below.

$$\underset{w}{min} \underset{t\sim p(T)}{E} \underset{x\sim p(D_t)}{E} L(\theta_t; O(w); x) \tag{16}$$

O is applying meta-dropout on w.

The optimization objective is below:

$$w* = \arg\underset{w}{max} \log p(O(w)|T) \tag{17}$$

In the meta-training, the formulas of optimization are:

$$w* = \arg\underset{w}{min} \underset{t\sim p(T)}{E} \underset{x\sim p(D_t)}{E} L^{meta}(\theta^{*(i)}(w), O(w)) \tag{18}$$

$$s.t.\theta^{*(i)}(w) = \arg\underset{\theta}{min} \underset{D_{ij}\sim p(D_i)}{E} L^{task}(\theta^{i}, O(w)) \tag{19}$$

Meta-dropout is applied on w. θ is task-specific that is not suitable for using meta-dropout. By using meta-dropout on w, the degree of over-fitting to the base classes can be relieved, thereby the model is easier to be adapted to novel classes.

Table 1: Comparison with several few-shot object detection methods on VOC2007 test set for novel classes of the three splits. Results are averaged over multiple seeds. **Red** / **black** indicate the state-of-the-art (SOTA) / the second best. * means applying our meta-dropout.

		split 1			split 2			split 3		
Model Type	Method/Shot	1	3	10	1	3	10	1	3	10
Episode-based	FR Kang et al. (2019)	14.8	26.7	47.2	15.7	22.7	**40.5**	**21.3**	28.4	**45.9**
	MetaDet Wang et al. (2019)	18.9	30.2	49.6	**21.8**	27.8	43	**20.6**	29.4	44.1
	Meta R-CNN Yan et al. (2019)	**19.9**	**35**	51.5	10.4	**29.6**	**45.4**	14.3	27.5	**48.1**
	Meta R-CNN (Our Impl.)	14.7	32.8	**51.9**	13.5	24.3	39.2	15.4	**33.8**	43.1
	Meta R-CNN*	**24.7**	**37.3**	**51.8**	**16.8**	**28.9**	40.1	20.2	**35.4**	45.7
Pre-train finetune	FRCN+ft Kang et al. (2019)	11.9	29	36.9	5.9	23.4	28.8	5.0	18.1	43.4
	FRCN+ft-full Yan et al. (2019)	13.8	32.8	45.6	7.9	26.2	39.1	9.8	19.1	45.1
	TFA Wang et al. (2020)	**25.3**	**42.1**	52.8	**18.3**	**30.9**	**39.5**	**17.9**	**34.3**	45.6
	TFA (Our Impl.)	22.4	40.3	**53.1**	15.6	26.7	37.4	16.9	32.3	**47.7**
	TFA*	**26.3**	**45.6**	**55.8**	**15.9**	**29.9**	**40.8**	**20.1**	**36.7**	**49**

4 Few-shot Object Detection

4.1 Datasets

We evaluate our methods on Pascal VOC Everingham et al. (2015; 2010) and MS COCO Lin et al. (2014). In PASCAL VOC, we adopt the common strategy Ren et al. (2016); Dai et al. (2016); Redmon & Farhadi (2017) that using VOC 2007 test set for evaluating while VOC 2007 and 2012 train/val sets are used for training. Following Yan et al. (2019), 5 out of its 20 object categories are selected as the novel classes, while keeping the remaining 15 ones as the base classes. We evaluate with three different novel/base splits from Yan et al. (2019), named as split 1, split 2, and split 3. By using the mean average precision (mAP) at 0.5 IoU threshold as the evaluation metric, we report the results on the official test set of VOC 2007. When using MS COCO, 20 out of 80 categories are reserved as novel classes, the rest 60 categories are used as base classes. The detection performance with COCO-style AP, AP50, and AP75 for K = 10 and 30 shots of novel categories are reported.

Table 2: Few-shot object detection performance on MS COCO. **Red** / **black** indicate the state-of-the-art (SOTA) / the second best.* means applying our meta-dropout.

	novel AP		novel AP50		novel AP75	
Method/Shot	10	30	10	30	10	30
FR Kang et al. (2019)	5.6	9.1	12.3	19	4.6	7.6
Meta R-CNN Yan et al. (2019)	8.7	**12.4**	**19.1**	**25.3**	6.6	10.8
TFA (Our Impl.)	**8.9**	12	16.2	21.2	**8.9**	**12.3**
TFA*	**9.7**	**12.8**	**17.5**	**22.3**	**9.8**	**13.1**

4.2 Implementation Details

Meta R-CNN which is a representative work of episode-based method is adopted as one of our baselines. We use ResNet-101 as the backbone with the structure of Faster R-CNN Ren et al. (2016).A simple yet effective pre-train finetune-based method Wang et al. (2020), named TFA, is also our baseline. Faster R-CNN is used as the detector and ResNet-101 with a Feature Pyramid Network Lin et al. (2017) is the backbone. During experiments, we find using dropblock Ghiasi et al. (2018) to implement meta-dropout is better than using normal dropout Srivastava et al. (2014) or spatial dropout Tompson et al. (2015).

4.3 Comparison with Baselines

Based on Meta R-CNN, we apply meta-dropout to get Meta R-CNN* with batch size 1. We also apply meta-dropout on TFA to get TFA*. Based on the official code, we re-implement Meta R-

CNN and TFA as our baselines denoted by Our Impl in Table. 1. Specifically, in episode-based methods, except the novel mAP of the 10-shot setting is comparable to the baseline, our Meta R-CNN* achieves more obvious improvement in all other settings. Notably, Meta R-CNN* can gain 10% improvement in the setting of split 1 with 1-shot. In pre-train finetune-based methods, our TFA* is able to obtain higher accuracy than the baseline in all settings. In general, our models get the highest improvement in the setting of 1-shot, followed by 3-shot, and the models get the least improvement in the 10-shot setting. When the number of shots becomes larger, the distribution of novel classes is easier to be obtained. With the decrease of the number of novel samples, meta-knowledge in meta-training which presents the power of generalization becomes more important. Therefore, using meta-dropout to improve the generalization power of meta-knowledge can help the models achieve higher improvement in the setting of fewer shots. Due to the few numbers of novel samples in the second training stage, we use multiple random seeds to get more stable results, as shown in Table. 1. While comparing to recent methods on Pascal VOC, we only use a certain seed to obtain the results, in order to be consistent with other methods, as shown in Table. 7 in Appendix.

Few-shot detection results of 10-shot and 30-shot setups for MS COCO are shown in Table. 2. Our methods TFA* can achieve about 1% gain in most metrics. It shows that the improvement of TFA is lower than that on PASCAL VOC, since MS COCO is a more challenging dataset.

4.4 Ablation Study

4.4.1 Meta-dropout

We apply meta-dropout on the meta-training stage and get the model as the initialization for meta-testing. In meta-testing, we do not use meta-dropout for adapting to the specific domain. When using Meta R-CNN as the baseline, using meta-dropout can achieve improvement in most settings presented in Table. 3. TFA is also utilized to prove the effect of meta-dropout, whose result is demonstrated in Table. 4. The experimental results show that meta-dropout can help TFA achieve higher performance in all settings. More ablation studies, including meta-dropout vs dropout, the implementation of meta-dropout, the specific locations of applying meta-dropout, and batch size, are shown in Appendix.

Table 3: Results of novel mAP on the split 1 of VOC 2007 test set by using meta-dropout on Meta R-CNN with batch size 1 and 4.

Setting / Shot (bs=1/4)	1	3	10
Meta R-CNN	23.3/14.7	**37.8**/32.8	49.3/**51.9**
+ meta-dropout	**24.7/16.6**	37.3/**32.9**	**51.8**/51.4

Table 4: Results of novel mAP on the split 1 of VOC 2007 test set by adopting meta-dropout on TFA.

Setting / Shot	1	3	10
TFA	22.4	40.3	53.1
+ meta-dropout	**26.3**	**45.6**	**55.8**

5 Few-shot Classification

5.1 Datasets

Caltech-UCSD Birds-200-2011 (CUB) Wah et al. (2011) is proposed for fine-grained classification, which contains 200 classes and 11,788 images in total. We follow the evaluation protocol of Hilliard et al. (2018) that 200 classes are divided into 100 base, 50 validation, and 50 novel classes, respectively. The mini-ImageNet Vinyals et al. (2016) that consists of 100 categories is a subset of ImageNet, and each class contains 600 images of size 84$\times$84. Follow the way of splitting the dataset in Finn et al. (2017); Ravi & Larochelle (2016); Snell et al. (2017); Sung et al. (2018), the selected 100 classes are divided into 64 training classes, 16 validation classes, and 20 test classes.

5.2 Implementation Details

We choose the Baseline++ Chen et al. (2019), which is an effective few-shot classification methods, as our baseline. During experiments, we follow the same setting in Chen et al. (2019). Specifically, Baseline++ is trained 200 epochs for the CUB dataset, and 400 epochs for the mini-ImageNet dataset.

5.3 Comparison with Baselines

We report the accuracy of few-shot classification in Table. 5, in which we build our Baseline++* by applying meta-dropout on Baseline++. When testing on the CUB dataset, we use dropblock with block size 7, which is applied on the last convolution layer. When testing on the mini-ImageNet dataset, normal dropout is used as meta-dropout that is applied on the flatten layer (1-dimensional feature) after the last convolution layer. The results demonstrate that our model is superior to the Baseline++ and outperforms other algorithms.

Table 5: Few-shot classification results for both the mini-ImageNet and CUB datasets.

		CUB		mini-ImageNet	
Method	backbone	1-shot	5-shot	1-shot	5-shot
MatchingNet Vinyals et al. (2016)	Conv-4	60.52±0.88	75.29±0.75	48.14±0.78	63.48±0.66
ProtoNet Snell et al. (2017)	Conv-4	50.46±0.88	76.39±0.64	44.42±0.84	64.24±0.72
MAML Finn et al. (2017)	Conv-4	54.73±0.97	75.75±0.76	46.47±0.82	62.71±0.71
RelationNet Sung et al. (2018)	Conv-4	62.34±0.94	77.84±0.68	49.31±0.85	66.60±0.69
Baseline Chen et al. (2019)	Conv-4	47.12±0.74	64.16±0.71	42.11±0.71	62.53±0.69
Baseline++ Chen et al. (2019)	Conv-4	60.53±0.83	79.34±0.61	48.24±0.75	66.43±0.63
Baseline++ (Our Impl.)	Conv-4	60.95±0.87	78.38±0.63	47.60±0.73	65.74±0.64
Baseline++*	Conv-4	**63.61±0.92**	**80.00±0.62**	**50.88±0.74**	**68.27±0.65**

5.4 Ablation Study

5.4.1 Meta-dropout

Based on the Baseline++ Chen et al. (2019) with batch size 16, we conduct experiments to explore the influence of meta-dropout, whose results are shown in Table. 6. We find using meta-dropout can significantly improve the performance of Baseline++ in almost all settings. More ablation studies, including meta-dropout vs dropout, the implementation of meta-dropout, the specific locations of applying meta-dropout, and batch size, are shown in Appendix.

Table 6: Results of Baseline++ on the CUB and mini-ImageNet datasets by applying meta-dropout.

		CUB		mini-ImageNet	
Method	backbone	1-shot	5-shot	1-shot	5-shot
Baseline++ (Our Impl.)	Conv-4	60.95±0.87	**78.38±0.63**	47.60±0.73	65.74±0.64
+meta-dropout	Conv-4	**62.71±0.87**	78.12±0.62	**50.47±0.72**	**68.20±0.65**
Baseline++ (Our Impl.)	ResNet-10	63.27±0.97	80.24±0.59	53.36±0.79	73.85±0.64
+meta-dropout	ResNet-10	**69.05±0.88**	**82.92±0.54**	**56.34±0.78**	**75.58±0.59**

6 Conclusion

In this paper, firstly, we introduce a unified meta-learning framework, which integrates two very different streams of few-shot learning, *i.e.*, the episode-based and pre-train finetune-based few-shot learning. Secondly, we propose a simple, general, and effective meta-dropout to improve the generalization power of meta-knowledge in our framework. Finally, we conduct extensive experiments on the challenging few-shot object detection and few-shot image classification tasks, in which our model demonstrates great superiority towards the current excellent few-shot learning methods. We believe that our framework can provide more important insights and the proposed meta-dropout will be widely used in few-shot learning.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pp. 3981–3989, 2016.

Hao Chen, Yali Wang, Guoyou Wang, and Yu Qiao. Lstd: A low-shot transfer detector for object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, and Xiaolong Wang. Meta-baseline: Exploring simple meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9062–9071, 2021.

Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *arXiv preprint arXiv:1605.06409*, 2016.

Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*, 2019.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

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

Qi Fan, Wei Zhuo, Chi-Keung Tang, and Yu-Wing Tai. Few-shot object detection with attention-rpn and multi-relation detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4013–4022, 2020.

Zhibo Fan, Yuchen Ma, Zeming Li, and Jian Sun. Generalized few-shot object detection without forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4527–4536, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *arXiv preprint arXiv:1810.12890*, 2018.

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4367–4375, 2018.

Spyros Gidaris and Nikos Komodakis. Generating classification weights with gnn denoising autoencoders for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 21–30, 2019.

Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3018–3027, 2017.

Nathan Hilliard, Lawrence Phillips, Scott Howland, Artëm Yankov, Courtney D Corley, and Nathan O Hodas. Few-shot learning with metric-agnostic conditional embeddings. *arXiv preprint arXiv:1802.04376*, 2018.

Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8420–8429, 2019.

Leonid Karlinsky, Joseph Shtok, Sivan Harary, Eli Schwartz, Amit Aides, Rogerio Feris, Raja Giryes, and Alex M Bronstein. Repmet: Representative-based metric learning for classification and few-shot object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5197–5206, 2019.

Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11–20, 2019.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

Bohao Li, Boyu Yang, Chang Liu, Feng Liu, Rongrong Ji, and Qixiang Ye. Beyond max-margin: Class margin equilibrium for few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7363–7372, 2021a.

Yiting Li, Haiyue Zhu, Yu Cheng, Wenxin Wang, Chek Sing Teo, Cheng Xiang, Prahlad Vadakkepat, and Tong Heng Lee. Few-shot object detection via classification refinement and distractor retreatment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15395–15403, 2021b.

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 *European conference on computer vision*, pp. 740–755. Springer, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Tsendsuren Munkhdalai and Hong Yu. Meta networks. *Proceedings of machine learning research*, 70:2554, 2017.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Bo Sun, Banghuai Li, Shengcai Cai, Ye Yuan, and Chi Zhang. Fsce: Few-shot object detection via contrastive proposal encoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7352–7362, 2021.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208, 2018.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pp. 181–209. Springer, 1998.

Pavel Tokmakov, Yu-Xiong Wang, and Martial Hebert. Learning compositional representations for few-shot recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6372–6381, 2019.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 648–656, 2015.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

Catherine Wah, Steve Branson, Pietro Perona, and Serge Belongie. Multiclass recognition and part localization with humans in the loop. In *2011 International Conference on Computer Vision*, pp. 2524–2531. IEEE, 2011.

Xin Wang, Thomas E Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. *arXiv preprint arXiv:2003.06957*, 2020.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Meta-learning to detect rare objects. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9925–9934, 2019.

Jiaxi Wu, Songtao Liu, Di Huang, and Yunhong Wang. Multi-scale positive sample refinement for few-shot object detection. In *European Conference on Computer Vision*, pp. 456–472. Springer, 2020.

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 *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9577–9586, 2019.

Weilin Zhang and Yu-Xiong Wang. Hallucination improves few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13008–13017, 2021.

A APPENDIX

A.1 FEW-SHOT OBJECT DETECTION

A.1.1 COMPARISON WITH STATE-OF-THE-ART METHODS

As shown in Table. 7, the results of all models are obtained on one selected seed. By applying our meta-dropout to FSCE, our method FSCE* can significantly outperform the baseline and achieve the state-of-the-art performance.

Table 7: Comparison with state-of-the-art few-shot object detection methods on VOC2007 test set for novel classes of the three splits. Results are obtained on one selected seed. **Red** / **Black** indicate state-of-the-art (SOTA) / the second best in the setting of single run. * is applying our meta-dropout.

		split 1			split 2			split 3		
Model Type	Method/Shot	1	3	10	1	3	10	1	3	10
Episode-based	CME Li et al. (2021a)	41.5	50.4	60.9	27.2	**41.4**	46.8	34.3	**45.1**	51.5
Pre-train finetune	MPSR Wu et al. (2020)	41.7	**51.4**	61.8	24.4	39.2	47.8	35.6	42.3	49.7
	FSCN Li et al. (2021b)	40.7	46.5	**62.4**	**27.3**	**40.8**	46.3	31.2	43.7	55.6
	Retentive R-CNN Fan et al. (2021)	42.4	45.9	56.1	21.7	35.2	40.3	30.2	43	50.1
	HallucFsDet Zhang & Wang (2021)	**47**	46.5	54.7	**26.3**	37.4	41.2	**40.4**	43.3	49.6
	FSCE Sun et al. (2021) (Our Impl.)	40.3	47.8	**62.2**	18.9	39.6	**49.6**	35.2	44.8	**56.1**
	FSCE*	**44.6**	**50.3**	61.7	24.6	**40.8**	**49.4**	**41.1**	**46.2**	**57.1**

A.1.2 META-DROPOUT VS DROPOUT

The implementation of dropout and our meta-dropout can be the same. However, meta-dropout is applied on the meta-knowledge in the meta-training stage, while dropout is applied on the meta-testing stage that is similar to the common one-stage training methods. Meta-dropout is for generalization power, while normal dropout is for specific tasks. In order to compare meta-dropout with dropout, we adopt meta-dropout, dropout, and meta-dropout&dropout on Meta R-CNN and TFA. When Meta R-CNN is the baseline, using dropblock to implement our meta-dropout and dropout, and applying it to group 4 of the backbone. When TFA is the baseline, normal dropout is used as our meta-dropout and dropout, and it is applied on the last convolution layer. From the results in Table. 8, only applying meta-dropout obtains higher accuracy than using dropout or both strategies in most settings. However, only using dropout on TFA can achieve the highest mAP in the 10-shot setting. The reason may be that dropout can help the pre-trained model get a better ability of fitting novel classes when fine-tuning with relatively more data. In 10-shot setting of Meta R-CNN, the baseline achieves the best performance, due to the characteristic of episode strategy when combined with more novel data.

Table 8: Results of novel mAP on the split 1 of VOC 2007 test set by adopting meta-dropout (M) and dropout (D).

	Meta R-CNN			TFA		
Setting / Shot	1	3	10	1	3	10
Baseline	14.7	32.8	**51.9**	22.4	40.3	53.1
+ M	**16.2**	**32.9**	51.4	**25.1**	**41.9**	53
+ D	13.3	32.5	50	23.1	41.8	**54.8**
+ M&D	14.5	32.5	48.8	25	41	50.5

Table 9: Results of mAP of novel classes on the split 1 of VOC 2007 test set by adopting different types of meta-dropout on Meta R-CNN and TFA.

	Meta R-CNN			TFA		
Setting / Shot	1	3	10	1	3	10
Baseline	14.7	32.8	**51.9**	22.4	40.3	53.1
+ dropout	14.3	31.3	50.2	25.1	41.9	53
+ dropblock	**16.2**	**32.9**	51.4	**25.9**	**42.5**	**54.5**

A.1.3 IMPLEMENTATION OF META-DROPOUT

We study the influence of the implementation of meta-dropout on performance. For example, our meta-dropout can be implemented by using normal dropout or dropblock. When using the Meta R-CNN framework, we apply meta-dropout on group 4 of the backbone, while meta-dropout is applied on the last convolution layer of TFA. The experimental results are shown in Table. 9. In general, using dropblock to implement our meta-dropout can get higher accuracy in novel classes.

A.1.4 SPECIFIC LOCATIONS OF APPLYING META-DROPOUT

Selecting Meta R-CNN and TFA as our baselines. In particular, dropblock is used to implement meta-dropout. From Table. 10, we conclude that applying meta-dropout on group 4 of the backbone is better than on group 3&4 with the setting of batch size 4. While using meta-dropout on group 3&4 achieves higher performance when batch size is 1. The results of applying meta-dropout on TFA are shown in Table. 11, in which using meta-dropout on group 3&4 gets the highest performance in most settings.The keep prob and block size are two important hyper-parameters in dropblock, which are set to 0.9 and 7 separately. Specifically, when using meta-dropout on group 3 or 4 of the backbone network, meta-dropout is applied to the last convolution layer of each bottleneck block.

Table 10: Results of novel mAP on the split 1 of VOC 2007 test set by adopting meta-dropout on different locations of Meta R-CNN.

Setting / Shot (bs=1/4)	1	3	10
Meta R-CNN	23.3/14.7	37.8/32.8	49.3/**51.9**
on last conv layer	20.3/15.6	31.6/32.2	46.3/50.8
on group 4	24.5/**16.2**	36/**32.9**	50/51.4
on group 3&4	**24.7**/16.1	**37.3**/32.5	**51.8**/50

Table 11: Results of novel mAP on the VOC 2007 test set by adopting meta-dropout on different locations of TFA.

	split 1			split 2		
Setting / Shot	1	3	10	1	3	10
TFA	22.4	40.3	53.1	15.6	26.7	37.4
on last conv layer	25.9	42.5	54.5	**17.3**	28.6	39.9
on group 4	**26.7**	44	54.2	14.3	28.3	39.3
on group 3&4	26.3	**45.6**	**55.8**	15.9	**29.9**	**40.8**

A.1.5 The influence of Meta-dropout on Base and Novel classes

Based on the original setting of TFA, we study the influence of meta-dropout on the performance of base and novel classes. The results shown in Table. 12 prove that our method can improve the accuracy of novel classes by a large margin without hurting the performance of base classes.

Table 12: Results of base and novel mAP on the split 1&2 of VOC 2007 test set by adopting meta-dropout on TFA.

	split 1			split2		
Base / Novel	1	3	10	1	3	10
TFA	79.8 / 37.2	79.1 / **44.4**	**79.4** / 53.4	79.8 / **21.6**	78.6 / 34.8	78.3 / 38.4
TFA*	**80 / 39.7**	**79.7** / 43.4	79.2 / **55.2**	**80.6** / 20.3	**78.8 / 36.6**	**78.7 / 41.8**

A.1.6 Batch Size

We explore the influence of batch size on Meta R-CNN and TFA. The experimental results are shown in Table. 13. Generally, on Meta R-CNN, as the batch size gets smaller, the performance is higher. Contrary to Meta R-CNN, as the batch size gets larger, the performance is higher in TFA, which is consistent with the previous experimental intuition.

Table 13: Results of overall and novel mAP on the split 1 of VOC 2007 test set by adopting different batch size (bs) on Meta R-CNN and TFA.

		Overall mAP		Novel mAP	
Methods	Bs / Shot	1	3	1	3
Meta R-CNN	1	**53.17**	**59.46**	**23.3**	**37.83**
	2	50.1	58.09	20.81	35.43
	4	43.9	55.73	14.74	32.83
TFA	4	55.4	58.2	17.2	30.8
	8	62.2	66.3	**23.5**	39.8
	16	**63.5**	**67.8**	22.4	**40.3**

A.2 Few-shot classification

A.2.1 Meta-dropout vs Dropout

We apply meta-dropout and dropout to show the importance of improving the generalization power of meta-knowledge. Using normal dropout as our meta-dropout and dropout. The results are shown in Table. 14. In the 5-shot setting of the CUB dataset, using meta-dropout is comparable to the Baseline++. However, applying meta-dropout can achieve the best performance in all other settings.

Table 14: Results of Baseline++ on the CUB and mini-ImageNet datasets by applying meta-dropout (M) and dropout (D).

	CUB		mini-ImageNet	
Method	1-shot	5-shot	1-shot	5-shot
Baseline++ (Our Impl.)	60.95±0.87	**78.38±0.63**	47.60±0.73	65.74±0.64
+ M&D	58.34±0.88	75.76±0.64	44.67±0.68	63.80±0.66
+ D	59.05±0.87	76.39±0.65	43.92±0.69	62.66±0.67
+ M	**62.71±0.87**	78.12±0.62	**50.47±0.72**	**68.20±0.65**

Table 15: Results on the CUB and mini-ImageNet datasets by applying different types of meta-dropout on Baseline++. *bl* is the block size in dropblock.

	CUB		mini-ImageNet	
Method	1-shot	5-shot	1-shot	5-shot
Baseline++ (Our Impl.)	60.95±0.87	78.38±0.63	47.60±0.73	65.74±0.64
dropblock (*bl*=3)	62.03±0.86	79.70±0.60	48.22±0.77	67.06±0.66
dropblock (*bl*=7)	**63.61±0.92**	**80.00±0.62**	47.60±0.71	67.25±0.62
dropout_conv	62.45±0.91	78.70±0.63	48.83±0.75	67.11±0.63
dropout_flatten	62.71±0.87	78.12±0.62	**50.47±0.72**	**68.20±0.65**

Table 16: Results on the CUB and mini-ImageNet datasets by applying meta-dropout on different locations of Baseline++.

	CUB		mini-ImageNet	
Method	1-shot	5-shot	1-shot	5-shot
Baseline++ (Our Impl.)	60.95±0.87	78.38±0.63	47.60±0.73	65.74±0.64
on last conv layer	62.45±0.91	**78.70±0.63**	48.83±0.75	67.11±0.63
on last flatten layer	**62.71±0.87**	78.12±0.62	**50.47±0.72**	**68.20±0.65**

A.2.2 Implementation of Meta-dropout

We study the way of implementing our meta-dropout by using dropblock with different block sizes and normal dropout with different locations. Due to the flatten layer is 1-dimensional, dropblock can only be used on the last convolution layer. Normal dropout can be applied on the last convolution or the last flatten layer. In Table. 15, the experimental results show that using dropblock with block size 7 is suitable for the CUB dataset, and applying normal dropout on the flatten layer achieves the best performance on the mini-ImageNet dataset.

A.2.3 Specific Locations of Applying Meta-dropout

For exploring the effect of the specific location of applying meta-dropout in the backbone, we use normal dropout as our meta-dropout, and apply it on the last convolution layer and the last flatten layer. The results in Table. 16 show that applying meta-dropout on the last 1-dimensional feature is better than on the last convolution layer.

A.2.4 Batch Size

Based on the Baseline++ Chen et al. (2019) with batch size 16, we conduct experiments to explore the influence of batch size, whose results are shown in Table. 17. We find using 32 as the batch size is the best choice for Baseline++.

Table 17: Results of Baseline++ on the CUB and mini-ImageNet datasets by adopting different batch size.

	CUB		mini-ImageNet	
Batch size	1-shot	5-shot	1-shot	5-shot
16	60.95±0.87	78.38±0.63	47.60±0.73	65.74±0.64
32	63.63±0.88	**78.96±0.63**	**50.21±0.75**	**68.23±0.67**
64	**63.67±0.88**	78.78±0.65	48.77±0.71	67.82±0.66
128	63.13±0.93	78.27±0.63	49.59±0.74	68.04±0.66