Task Discrepancy Maximization for Fine-grained Few-Shot Classification

SuBeen Lee, WonJun Moon, Jae-Pil Heo*
Sungkyunkwan University
{leesb7426, wjun0830, jaepilheo}@skku.edu

Abstract

Recognizing discriminative details such as eyes and beaks is important for distinguishing fine-grained classes since they have similar overall appearances. In this regard, we introduce Task Discrepancy Maximization (TDM), a simple module for fine-grained few-shot classification. Our objective is to localize the class-wise discriminative regions by highlighting channels encoding distinct information of the class. Specifically, TDM learns task-specific channel weights based on two novel components: Support Attention Module (SAM) and Query Attention Module (QAM). SAM produces a support weight to represent channel-wise discriminative power for each class. Still, since the SAM is basically only based on the labeled support sets, it can be vulnerable to bias toward such support set. Therefore, we propose QAM which complements SAM by yielding a query weight that grants more weight to object-relevant channels for a given query image. By combining these two weights, a class-wise task-specific channel weight is defined. The weights are then applied to produce task-adaptive feature maps more focusing on the discriminative details. Our experiments validate the effectiveness of TDM and its complementary benefits with prior methods in fine-grained few-shot classification.

1. Introduction

With the advancement of deep learning, it has achieved remarkable performance beyond humans in various downstream tasks [5, 10]. However, there is a strong assumption that numerous labeled images should exist to achieve such performance. If the number of labeled images is insufficient, it shows drastic degradation in the performance [3, 8, 38]. To resolve such degradation from a shortage of labeled images and reduce the cost of labeling, the computer vision community recently paid more attention to few-shot classification [8, 34, 38]. Briefly, the goal of few-shot classification is to train a model with high adaptability to novel classes. To achieve this goal, the episodic learning strategy is mainly used, where each episode consists of sampled categories from the dataset. Furthermore, each class has a support set for training and a query set for evaluation.

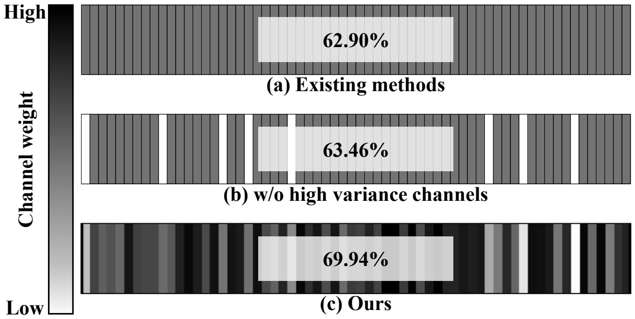


Figure 1. Effect of the channel weight in the CUB dataset. (a) Existing methods treat channels of feature maps equally. In such case, high variance channels within a class highly likely to disturb the classification task, where channel variance represents the channel-wise variances of feature maps of the same class – intuitively, the instances of the same class having similar features at a channel lead a low channel variance for the corresponding channel. This is mainly because it is hard to make a consensus among features for classification criteria. (b) However, in fine-grained datasets, simply removing high variance channels for each class shows marginal improvements. It is because classes share similar features, e.g., feather, and wings in CUB dataset, and thus channels with a low variance may not be discriminative. Therefore, in fine-grained datasets, we should grant different weights to channels depending on whether each channel reflects distinct characteristics. (c) TDM produces per-class channel weight by discovering discriminative channels for each class in the episode. Note that, the numbers in boxes are classification accuracies.

The stream of metric-based learning is a promising direction for the few-shot classification. These methods [16, 34, 35, 38] learn a deep representation with a predefined metric or online-trained metric. Specifically, the inference for a query is performed based on the distances among support and query sets under such metric.

However, the features of a novel class extracted by a model trained on the base classes hardly form a tight cluster, since the feature extractor is highly sensitive and activates the semantically discriminative variations in the distribution of base classes [32, 46]. To alleviate this, recent methods

[*] Corresponding author

utilize primitive knowledge [20,46] or propose task-dynamic feature alignment strategies [7, 12, 14, 33, 42, 44, 45]. Among two strategies, task-dynamic feature alignment methods are being spotlighted. The task-dynamic feature alignment methods can be further divided into two main streams: spatial alignment and channel alignment. The spatial alignment methods [7, 12, 14, 42, 42, 44] aim to resolve the spatial mismatch between key features on the feature maps of different instances. On the other hand, the channel alignment methods [14, 33, 44, 45] modify feature maps to better represent the semantic features for novel classes.

Although these alignment methods are shown to be effective on the general few-shot classification task, they achieved insignificant gains for fine-grained datasets. This is mainly because they only focus on exploiting features that describe novel objects, which may not be discriminative in such tasks. Indeed, localizing discriminative details is important in fine-grained classification, since categories share similar overall appearances [6, 9, 27, 50]. Therefore, distinct clues for each category also should be discovered for fine-grained few-shot classification. In Fig. 1 (c), we verify that localizing discriminatory details of the object through channel weights is effective for fine-grained few-shot task.

In this context, we introduce novel Task Discrepancy Maximization (TDM), a module that localizes discriminative regions by weighting channels per class. TDM highlights the channels that represent discriminative regions and restrains the contributions of other channels based on class-wise channel weight. Specifically, TDM is composed of two components: Support Attention Module (SAM) and Query Attention Module (QAM). Given a support set, SAM outputs a support weight per class that presents high activations on discriminative channels. On the other hand, QAM is fed with the query set to produce a query weight per instance. The query weight is to highlight the object-relevant channels. To infer these weights, the relation between each feature map and the average channel pooled features are considered. Note that, the channel pooled average feature map has the spatial information of the object [22, 43] as described in Fig. 2. Therefore, channels are highly likely to represent objects when they are similar to spatially averaged feature map. By combining two weights computed from our sub-modules, a task-specific weight is finally defined. Consequently, the task-specific weight is utilized to produce task-adaptive feature maps.

Our main contributions are summarized as follows:

- We propose a novel feature alignment method, TDM, to define the class-wise channel importance, for fine-grained few-shot classification.

- Our proposed TDM is highly applicable to prior metric-based few-shot classification models.

- When combined with the recent few-shot classification models, the TDM achieves the state-of-the-art performance in fine-grained few-shot classification task.

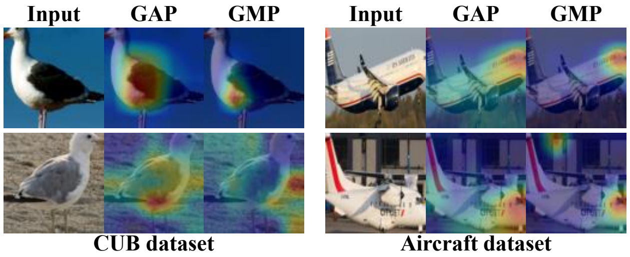


Figure 2. Visualization of pooling results. Each column shows the locations where each pooling method focuses on the image. The second and third columns visualize the results of average pooling and the max pooling, respectively. GAP tends to concentrate on the object parts in the images, while GMP is often out of focus.

2. Related Works

2.1. Few-Shot Classification

The methods of few-shot classification can be divided into two main streams: optimization- and metric-based. The concept of optimization-based methods was introduced in MAML [8] to learn good initial conditions that can be easily adapted. Meta-LSTM [31] adopts an LSTM-based meta-learner which is not only for the general initial point but also for effective fine-tuning. MetaOptNet [19] employs convex base learners, and provides a differentiation process for end-to-end learning. The optimization-based methods show comparable performance, but they need online updates for novel classes.

The metric-based methods learn deep representations by utilizing a predefined [16, 34, 38] or online-trained metric [35]. MatchNet [38] uses an external memory module which augments neural networks and infers categories of the query set by the cosine similarity. ProtoNet [34] forms prototypes with a mean feature of each class in support set, and exploits them for computing the distance between a query to each class. RelationNet [35] utilizes the distance metric learned by a model instead of the predefined metric.

Metric-based methods generally learn to reduce distances among instances within a class, and we have the same goal since TDM is a module for them. However, TDM enables to compute the distances based on adaptive channel weights by identifying discriminative channels dynamically, while prior techniques treat all the channels equally.

2.2. Feature Alignment

Feature alignment methods can be categorized into spatial and channel alignments. The spatial alignment methods [7, 12, 14, 42, 44, 47] assert that object location differences in the support and query set cause the performance degradation. CAN [12] computes cross attention maps by calculating

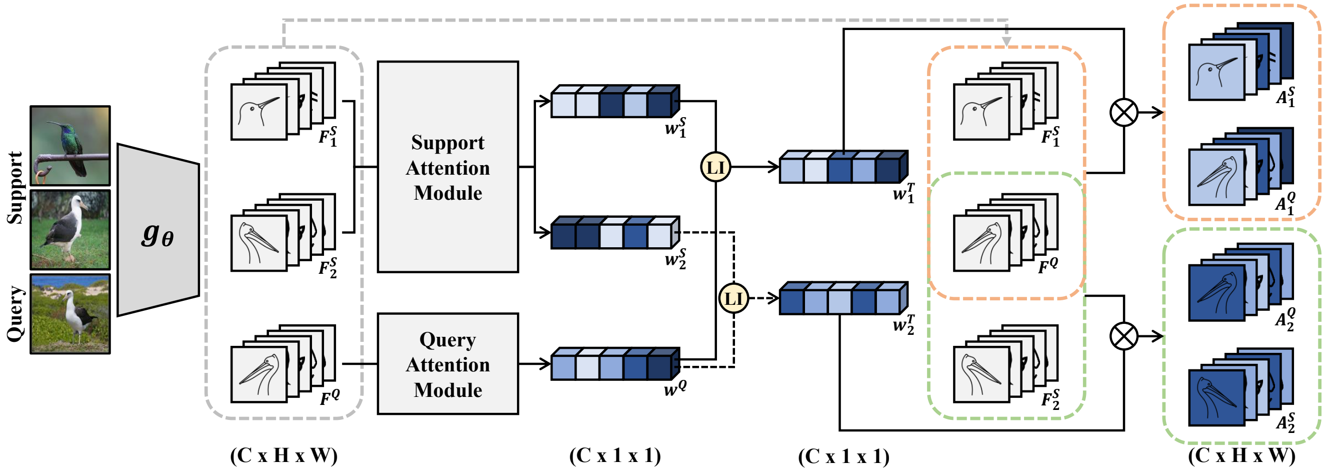


Figure 3. Overview of our approach. The Task Discrepancy Maximization consists of two sub-modules. Each sub-module takes feature maps F and generates channel weights w. The support attention module utilizes feature maps of the support instances as input and finds discriminative channels for each class. Then, it produces a support weight w_i^S for i-th category where the weight holds high values in those channels. On the other hand, the query attention module is fed the query instance and discovers the object-relevant channels of the query. Then, a query weight w^Q from the query attention module emphasizes specific channels with object information about the query. These weights from two sub-modules are integrated by a linear combination to define a task weight w_i^T for each i-th category. Finally, task-adaptive feature maps which concentrate on the discriminative regions are obtained by multiplying the task weights to original feature maps.

correlation for each pair of the classes and query feature maps to highlight the common regions to locate the object. CTX [7] finds a coarse spatial correspondence between the query instance and the support set by the attention [2] to produce a query-aligned prototype per each class. FRN [42] reconstructs the feature maps of the support set to the query instance by exploiting a closed-form solution of the ridge regression.

The channel alignment methods [14, 33, 44, 45] manipulate feature maps to be capable to represent novel classes. FEAT [45] increases the distances among classes of the support set by adopting the transformer [25, 37]. DMF [44] aligns feature maps of the query instance by the dynamic meta-filter which has both position and channel-specific support knowledge. RENet [14] transforms a feature map with self-correlation capturing structural patterns of each image.

TDM also deals with the feature alignment. Unlike existing methods that typically consider a pairwise relationship between the support image and the query image, TDM considers the entire task.

3. Method

The overall architecture of our method is illustrated in Fig. 3. Given an episode consisting of the support and query instances, feature maps are first computed by the feature extractor. However, the feature maps are not optimal for each episode since the feature extractor is trained to find discriminative features for classifying base classes [32, 45, 46]. TDM transforms the feature maps by exploiting task-specific weights representing channel-wise discriminative power for a specific task. As a result, we aim to focus on the discriminatory details by refining the original feature maps into task-adaptive feature maps. In this section, we introduce the components of TDM and their purpose. First, we formulate the problem in Sec. 3.1. In Sec. 3.2, we define two representative scores to produce channel weights. Then, with these scores, we describe two modules of the TDM: SAM and QAM in Sec. 3.3 and Sec. 3.4, respectively. Finally, TDM is described in Sec. 3.5 with the discussion in Sec. 3.6.

3.1. Problem Formulation

In standard few-shot classification, we are given two datasets: meta-train set $D_{base} = \{(x_i, y_i), y_i \in C_{base}\}$ and meta-test set $D_{novel} = \{(x_i, y_i), y_i \in C_{novel}\}$. C_{base} and C_{novel} represent base classes and novel classes, respectively, where they do not overlap ($C_{base} \cap C_{novel} = \phi$). Generally, training and testing of few-shot classification are composed of episodes. Each episode consists of randomly sampled N classes and each class is composed of K labeled images and U unlabeled images, *i.e.*, N-way K-shot episode. The labeled images are called the support set $S = \{(x_j, y_j)\}_{j=1}^{N \times K}$, and the unlabeled images are named the query set $Q = \{(x_j, y_j)\}_{j=1}^{N \times U}$, while two sets are disjoint ($S \cap Q = \phi$). The support and query sets are utilized for learning and testing, respectively.

3.2. Channel-wise Representativeness Scores

For each pair of i-th class and c-th channel, we define two channel-wise representativeness scores; intra score $R_{i,c}^{\text{intra}}$, and inter score $R_{i,c}^{\text{inter}}$. Prior to explaining scores, we first define feature maps F of the support and query instance as follows:

$$\begin{aligned} F^S_{i,j} &= g_\theta(x^S_{i,j}) \\ F^Q &= g_\theta(x^Q), \end{aligned} \tag{1}$$

where $x^S_{i,j}$ is j-th instance of i-th class in the support set, x^Q is the query instance, and g_θ is our feature extractor parameterized by θ. Each feature map $F \in \mathbb{R}^{C\times H\times W}$ where C, H, W denote the number of channels, height, and width, respectively. Additionally, we utilize a prototype [34] as the representative of each class. The prototype F^P_i for i-th class is computed as follows:

$$F^P_i = \frac{1}{K}\sum_{j=1}^{K} F^S_{i,j}. \tag{2}$$

For each class, we first compute a mean spatial feature to represent a salient object regions. When the c-th channel of the prototype for i-th class is denoted as $f^P_{i,c} \in \mathbb{R}^{H\times W}$, the corresponding mean spatial feature M^P_i is computed as follows:

$$M^P_i = \frac{1}{C}\sum_{j=1}^{C} f^P_{i,j}. \tag{3}$$

Based on this, we further compute the channel-wise representativeness score defined within a class, $R^{\text{intra}}_{i,c}$, for c-th channel of i-th class as follows:

$$R^{\text{intra}}_{i,c} = \frac{1}{H\times W} \parallel f^P_{i,c} - M^P_i \parallel^2 . \tag{4}$$

The score indicates how well the highly activated regions on the channel are matched with the class-wise salient area represented by the mean spatial feature. On the other hand, the channel-wise representativeness score across classes, $R^{\text{inter}}_{i,c}$, for c-th channel of i-th class is defined as follows:

$$R^{\text{inter}}_{i,c} = \frac{1}{H\times W} \min_{1\leq j\leq N, j\neq i} \parallel f^P_{i,c} - M^P_j \parallel^2 . \tag{5}$$

It represents how much c-th channel contains the distinct information of each class. Intuitively, the channel is more discriminative when it has the smaller intra score and the larger inter score. We utilize both scores to define channel weights.

3.3. Support Attention Module (SAM)

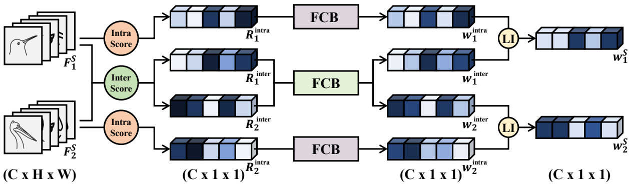


Figure 4. Schemetic illustration of Support Attention Module.

For each class, our support attention module (SAM) takes the class prototypes as input, and first compute two channel-wise representativeness scores based on Eq. (4) and Eq. (5). We transform those two scores to two weights, w^{intra}_i and w^{inter}_i, for i-th class as follows:

$$\begin{aligned} w^{\text{intra}}_i &= b^{\text{intra}}\left(R^{\text{intra}}_i\right) \\ w^{\text{inter}}_i &= b^{\text{inter}}\left(R^{\text{inter}}_i\right), \end{aligned} \tag{6}$$

where b^{intra} and b^{inter} are different fully-connected blocks. The architectures of the blocks are reported in Tab. 1.

The support weight vector w^S_i for i-th class is defined by a linear combination of the corresponding two weights, w^{intra}_i and w^{inter}_i, with a balancing hyperparameter α as follows:

$$w^S_i = \alpha w^{\text{intra}}_i + (1-\alpha) w^{\text{inter}}_i,\ \alpha \in [0, 1]. \tag{7}$$

The support weight vector for i-th class emphasizes discriminative channels of i-th class while suppressing channels that corresponds to the common information shared throughout classes in the episode. When we multiply the support weight vector for i-th class, w^S_i, to the feature maps, the instances of i-th class should be gathered, while other class instances become separated from the i-th class.

3.4. Query Attention Module (QAM)

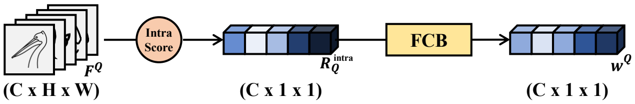


Figure 5. Schemetic illustration of Query Attention Module

Although the support weight vectors are useful to distinguish the class-wise informative channels, we are also encouraged to utilize the query set to overcome the data scarcity in the few-shot learning. To utilize the query set for the complementary benefits with SAM, we propose the query attention module (QAM). Since we do not have label information for a query instance, QAM only exploits the relationship among channels within an instance. By utilizing the mean spatial feature M^Q defined by the channel-wise mean of query's feature map F^Q, we compute the channel-wise representativeness score of query instance, R^{intra}_Q, for c-th channel as follows:

$$R^{\text{intra}}_Q = \frac{1}{H\times W} \parallel f^Q_c - M^Q \parallel^2, \tag{8}$$

where f^Q_c denotes c-th channel of query's feature map. Then, the query weight w^Q is produced by passing the intra score R^{intra}_Q to the fully connected block b^Q as described in Tab. 1.

$$w^Q = b^Q\left(R^{\text{intra}}_Q\right). \tag{9}$$

The query weight vector highlights object-relevant channels of the query instance while restraining others. Therefore, query weight vector assists the model to focus on object-related information.

Fully Connected Block	
Layer	Output Size
Input	B × C
Fully Connected Layer	B × 2C
Batch Normalization	B × 2C
ReLU	B × 2C
Fully Connected Layer	B × C
1 + Tanh	B × C

Table 1. The architecture of the fully connected blocks. When used in the support attention module, the batch size B is the number of categories which is comprising an episode. In the query attention module, B is the number of queries.

3.5. Task Discrepancy Maximization (TDM)

As the support and query weight vectors computed by SAM and QAM are complementary in their purposes, we use them to produce a task weight vector. Specifically, the task weight vector w_i^T for i-th class is defined by a linear combination of the corresponding support and query weight vectors, w_i^S and w^Q, with a hyperparameter β as follows:

$$w_i^T = \beta w_i^S + (1-\beta) w^Q, \quad \beta \in [0,1]. \tag{10}$$

Then, the feature maps of all the support and query instances are transformed with the task weight vector. Specifically, each feature map $F \in \mathbb{R}^{C\times H\times W}$ is transformed to a task-adaptive feature map A with channel-wise scaling by the task weight vector $w_i^T \in \mathbb{R}^C$ as follows:

$$A = w_i^T \odot F = \left[w_{i,1}^T f_1, w_{i,2}^T f_2, ..., w_{i,C}^T f_C\right], \tag{11}$$

where $w_{i,j}^T$ is a scalar value at j-th dimension of the vector w_i^T, and the c-th channel of the feature map F is denoted by $f_c \in \mathbb{R}^{H\times W}$.

The feature maps of the support instances of i-th class are transformed by its corresponding task weight vector w_i^T. However, since the label of the query is not available, we apply the task weight vector for i-th class, w_i^T, to the query, when we are testing the query for the i-th class. As a result, based on Eq. (11), we compute task adaptive feature maps of the support instances $A_{i,j}^S$ and the query instance A_i^Q transformed by the task weight vector w_i^T as follows:

$$\begin{aligned} A_{i,j}^S &= w_i^T \odot F_{i,j}^S \\ A_i^Q &= w_i^T \odot F^Q, \end{aligned} \tag{12}$$

where i, j denote the class index and instance index within the class, respectively.

For instance, when TDM is applied to the ProtoNet [34], the inference process is done by the following criteria:

$$p_\theta(y=i|x) = \frac{\exp(-d(A_i^S, A_i^Q))}{\sum_{j=1}^N \exp(-d(A_j^S, A_j^Q))}, \tag{13}$$

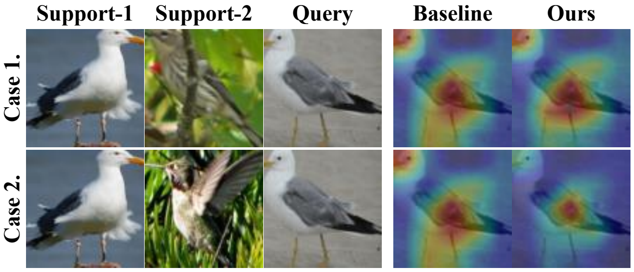


Figure 6. 2D-aggregated feature activation on 2-way 1-shot. (Case 1) If beaks and wings are not similar between species, TDM regards both beaks and wings to be discriminative. (Case 2) However, when birds have similar beaks, TDM considers only wings as a discriminative part.

where d indicates the distance metric, and A_i^S is the prototype computed by the adaptive feature maps of support instances in i-th class.

3.6. Discussion

Commonly, it is known that the feature map containing various information about the object is beneficial for a general dataset [13, 21, 23]. However, in a fine-grained dataset, those feature maps are often harmful to infer the categories. Instead, it is advantageous to focus only on the discriminative parts since the categories share common overall apperance [6, 9, 27, 50]. Moreover, unlike general fine-grained classification where the discriminative region of each category is almost constant, the distinct region of each class in few-shot setting may vary depending on the contents of the episode. Therefore, the key point of fine-grained few-shot classification is dynamically discovering the discriminative regions of each class based on very small number of instances. In Fig. 6, we observe that the baseline model treats all characteristics equally regardless of the composition of each episode. However, TDM adaptively allows the model to highlight regions expected as discriminative parts and suppress other regions. Thus, TDM is a specialized module for fine-grained few-shot classification.

4. Experiments

In this section, we evaluate TDM on standard fine-grained classification benchmarks. To verify high adaptability of TDM, we apply it to various existing methods: ProtoNet [34], DSN [33], CTX [7], and FRN [42]. For a fair comparison, we reproduce each baseline model with the same hyperparameter and implementation details regardless of whether TDM is attached or not. In each table, † indicates the reproduced version of the baseline model. While TDM generally exploits the prototype [34] defined in Eq. (2) for computing the intra and inter score, in CTX, it utilizes a query-aligned prototype [7].

Dataset	C_{all}	C_{train}	C_{val}	C_{test}
CUB-200-2011	200	100	50	50
Aircraft	100	50	25	25
meta-iNat	1135	908	-	227
tiered meta-iNat	1135	781	-	354
Stanford-Cars	196	130	17	49
Stanford-Dogs	120	70	20	30
Oxford-Pets	37	20	7	10

Table 2. The splits of datasets. While C_{all} is the number of total classes, C_{train}, C_{val}, C_{test} are the number of training, validation, and test classes, respectively. The classes of subsets are disjoint.

4.1. Datasets

We utilize seven benchmarks for few-shot classification: CUB-200-2011, Aircraft, meta-iNat, tiered meta-iNat, Stanford-Cars, Stanford-Dogs, and Oxford-Pets. The data split for each dataset is provided in Tab. 2.

CUB-200-2011 [39] is an image dataset with 11,788 photos of 200 bird species. This benchmark can be used in two ways: raw form [3] or preprocessed form by a human-annotated bounding box [45, 47]. For a fair comparison, we utilizes both settings. Following [3], we split this benchmark and our split is the same with [42].

Aircraft [28] contains 10,000 airplane images of 100 models. The main challenge of this benchmark is the similarity by the symbol of the airline. Although the types of aircraft are different, the symbol can be equivalent when they belong to the same airline, making it more difficult. Following [42], we preprocess all images of this benchmark based on the bounding box and divide the dataset.

meta-iNat [36, 41] is a realistic, heavy-tailed benchmark for few-shot classification. It contains 1,135 animal species spanning 13 super categories, and the number of images in each class is imbalanced with a range between 50 and 1000 images. We follow the dataset split introduced in [41]. While a full 227-way evaluation scheme, that each episode consists of all testing categories at once, is adopted in [41], we employ a standard 5-way few-shot evaluation scheme following [42].

tiered meta-iNat [41] is comprised of the same images with meta-iNat. However, this benchmark divides the dataset by super categories. The super categories of test dataset are Insects and Arachnids, while the super categories of train dataset are Plant, Bird, Mammal, and etc. Therefore, a large domain gap exists between the train and the test classes. As following [42], we also employ a standard 5-way few-shot evaluation scheme in this benchmark.

Stanford Cars [18] contains 16,185 images of 196 classes of cars. Classes are typically at the level of Year, Brand and Model name, e.g., 2012 Tesla Model S and 2012 BMW M3 coupe. This dataset was introduced by [24] for the few-shot classification. Likewise, we adopt the same data split with [24].

Model	Conv-4		ResNet-12	
	1-shot	5-shot	1-shot	5-shot
MatchNet [38, 45, 47]	67.73	79.00	71.87	85.08
ProtoNet [34, 45, 47]	63.73	81.50	66.09	82.50
FEAT* [45]	68.87	82.90	73.27	85.77
DeepEMD [47]	-	-	75.65	88.69
RENet [14]	-	-	79.49	91.11
ProtoNet† [34]	62.90	84.13	78.99	90.74
+ TDM	69.94	86.96	79.58	91.28
DSN† [33]	72.09	85.03	80.51	90.23
+ TDM	73.38	86.07	81.33	90.65
CTX† [7]	72.14	87.23	80.67	91.55
+ TDM	**74.68**	88.36	83.28	92.74
FRN† [42]	73.24	88.33	83.16	92.42
+ TDM	74.39	**88.89**	**83.36**	**92.80**

Table 3. Performance on CUB using bounding-box cropped images as input. "$*$" denotes reproduced one in RENet. Confidence intervals for our implemented model are all below 0.23.

Model	Backbone	1-shot	5-shot
Baseline$^\flat$ [3]	ResNet-18	65.51±0.87	82.85±0.55
Baseline++$^\flat$ [3]	ResNet-18	67.02±0.90	83.58±0.54
MatchNet$^\flat$ [3, 38]	ResNet-18	73.42±0.89	84.45±0.58
ProtoNet$^\flat$ [3, 34]	ResNet-18	72.99±0.88	86.65±0.51
MAML$^\flat$ [3, 8]	ResNet-18	68.42±1.07	83.47±0.62
RelatioNet$^\flat$ [3, 35]	ResNet-18	68.58±0.94	84.05±0.56
S2M2$^\flat$ [29]	ResNet-18	71.43±0.28	85.55±0.52
Neg-Cosine$^\flat$ [26]	ResNet-18	72.66±0.85	89.40±0.43
Afrasiyabi *et al.*$^\flat$ [1]	ResNet-18	74.22±1.09	88.65±0.55
ProtoNet† [34]	ResNet-12	78.58±0.22	89.83±0.12
+ TDM	ResNet-12	79.11±0.22	90.83±0.11
DSN† [33]	ResNet-12	80.47±0.20	89.92±0.12
+ TDM	ResNet-12	80.58±0.20	89.95±0.12
CTX† [7]	ResNet-12	80.95±0.21	91.54±0.11
+ TDM	ResNet-12	83.45±0.19	92.49±0.11
FRN† [42]	ResNet-12	83.54±0.19	92.96±0.10
+ TDM	ResNet-12	**84.36±0.19**	**93.37±0.10**

Table 4. Performance on CUB using raw images as input. "$\flat$" denotes larger backbones than *ResNet-12*.

Stanford Dogs [15] is also introduced by [24] for fine-grained few-shot classification. This dataset contains 20,580 images of 120 breeds of dogs around the world. Our split is the same with [24].

Oxford Pets [30] is comprised of 37 pet categories with roughly 200 images for each class. Since the lack of training images generally causes overfitting, the generalization capability is essential to attain high accuracy in this benchmark. To our knowledge, this benchmark dataset has never been used for few-shot classification. Therefore, we randomly divide this dataset by referring to the split ratio of the other datasets. We report the split information in the supplementary material.

4.2. Implementation Details

Architecture. We adopt common protocols from recent few-shot classification works [4, 11, 17, 48, 49]; we employ Conv-4 and ResNet-12. While both backbone networks accept an image of size 84×84, the size of feature maps is different according to the backbone network. ResNet-12 yields a feature map with size 640×5×5 while Conv-4 offers 64×5×5 shape of the feature map. For our proposed TDM, we additionally utilize fully-connected layer blocks where the size of blocks are proportional to the dimension of channels of the feature maps as described in Tab. 1.

Training Details. Following existing methods [3, 40, 42, 45, 47], we use standard data augmentation techniques including random crop, horizontal flip, and color jitter. The α, β in Eq. (7), Eq. (10) are fixed to 0.5 and other parameters are adopted from [42] which is our baseline model. More details are described in the supplementary material. To prevent overfitting, we add random noise between -0.2 and 0.2 to the task weight for each category from TDM.

Evaluation Details. For the N-way K-shot, we conduct few-shot classification on 10,000 randomly sampled episodes where it contains 16 queries per class. We report average classification accuracy with 95% confidence intervals.

4.3. Comparison to Existing Methods

CUB-200-2011 results. Tab. 3 and Tab. 4 reports the results of TDM and baseline few-shot classification methods. Although some cases do not surpass confidence intervals, our TDM consistently improves the performance of baselines models in all cases, and achieves state-of-the-art scores regardless the depth of the backbone network. Especially, TDM improves more than 7% on ProtoNet in the 1-shot scenario of CUB with Conv-4.

Aircraft results. As shown in Tab. 5, TDM improves the performances of the baseline models in all cases. The increases are beyond the confidence intervals regardless of the type of baseline models and the number of labeled images. As a result, we attain top accuracy scores for all benchmarks by large margin. Specifically, TDM boosts the performance of CTX [7] up to 7% on the Conv-4 network. These results demonstrates the effectiveness of TDM.

meta-iNat results. We validate the generalization ability of TDM in this benchmark. Meta-iNat is vulnerable to overfitting since the validation set does not exist. However, as shown in Tab. 6, TDM is not only robust to overfitting issues but also powerful in generalization capability. Consequently, TDM shows consistent improvements over the baselines, particularly on ProtoNet where TDM assists them to retain competitive results compared to state-of-the-art methods.

Model	Conv-4		ResNet-12	
	1-shot	5-shot	1-shot	5-shot
ProtoNet† [34]	47.37	68.96	67.28	83.21
+ TDM	50.55	71.12	69.12	**84.77**
DSN† [33]	52.22	68.75	70.23	83.05
+ TDM	53.77	69.56	**71.57**	83.65
CTX† [7]	51.58	68.12	65.53	79.31
+ TDM	**55.15**	70.45	69.42	83.25
FRN† [42]	53.12	70.84	69.58	82.98
+ TDM	54.21	**71.37**	70.89	84.54

Table 5. Performance on Aircraft. Confidence intervals for our implemented model are all below 0.25.

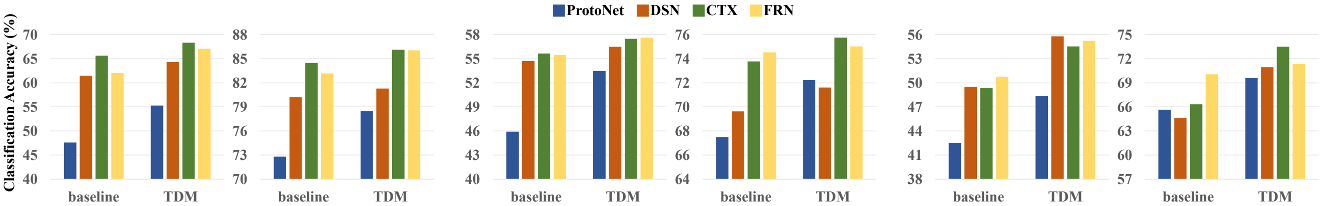
Model	meta-iNat		tiered meta-iNat	
	1-shot	5-shot	1-shot	5-shot
ProtoNet† [34]	55.37	76.30	34.41	57.60
+ TDM	61.82	79.95	38.30	61.18
DSN† [33]	60.06	76.15	40.83	58.34
+ TDM	61.87	78.07	41.00	58.66
CTX† [7]	60.80	78.57	42.24	60.54
+ TDM	63.26	80.75	43.90	62.29
FRN† [42]	61.98	80.04	43.95	**63.45**
+ TDM	**63.97**	**81.60**	**44.05**	62.91

Table 6. Performance on meta-iNat and tiered meta-iNat using Conv-4 backbones. Confidence intervals for our implemented model are all below 0.23.

tiered meta-iNat results. We further validate the generalization capability of TDM in a more difficult configuration where the super categories of train set and test set do not overlap. Specifically, TDM enhances the performance in most configurations and also accomplishes the best performance in the 1-shot scenario with FRN. For the slight decrease in a 5-shot scenario, λ, the learnable parameter in FRN, is responsible. In general, large λ shows good performance when a domain gap exists but TDM restrains the λ to be relatively small. We think that this is because TDM assists the classifier to focus on discriminative channels [42].

Stanford Cars, Standford Dogs and Oxford Pets results. Unlike previous benchmarks, these datasets were not evaluated in our baseline models [7, 33, 34, 42]. To further validate the effectiveness of TDM, we additional conduct experiments on those fine-grained datasets with Conv-4. As reported in Fig. 7, TDM is capable to improve the performance that the confidence interval does not overlap in all cases regardless of the baseline models. In detail, TDM shows performance boosts in which their accuracies are 4.44 and 3.27 points higher than the baseline at the 1-shot and 5-shot scenarios, respectively.

Throughout the extensive experiments on seven benchmark datasets, we clearly validate the strength of TDM in fine-grained few-shot classification. To summarize, we improve the performance of the baseline models in all benchmarks datasets, achieving the state-of-the-art results except the one case: FRN on the tiered meta-iNat 5-shot scenario.



(a) Standford Cars (b) Standford Dogs (c) Oxford Pets

Figure 7. Accuracies on additional datasets. The left and right graphs for each dataset show 1-shot accuracies and 5-shot accuracies, respectively. The left side of the graph reports the performance of baseline models, while the right side shows performance with TDM. The baseline methods are differentiated with colors.

SAM	QAM	CUB_cropped		Aircraft	
		1-shot	5-shot	1-shot	5-shot
-	-	62.90	84.13	47.37	68.96
✓	-	68.53	85.95	49.45	69.33
-	✓	65.11	84.82	48.96	70.85
✓	✓	**69.94**	**86.96**	**50.55**	**71.12**

Table 7. Effects of SAM and QAM.

TDM	Pooling	CUB_cropped		Aircraft	
		1-shot	5-shot	1-shot	5-shot
-	-	62.90	84.13	47.37	68.96
✓	Max	67.23	86.73	50.16	**71.32**
✓	Avg	**69.94**	**86.96**	**50.55**	71.12

Table 8. Effects of Pooling functions. Top row is the baseline result.

TDM	CUB_cropped		Aircraft	
	1-shot	5-shot	1-shot	5-shot
-	68.69	82.89	48.36	63.45
✓	**70.47**	**84.34**	**49.21**	**66.26**

Table 9. Effectiveness of TDM with the cosine distance

5. Ablation Study

We conduct ablation study with ProtoNet [34] on the Conv-4 backbone using CUB_cropped and Aircraft datasets.

5.1. Effect of Submodules

Tab. 7 reports the effects of sub-modules of TDM. We observe that both SAM and QAM consistently improve the classification accuracies. As the second and fourth rows show, SAM improves the baseline by a large gain up to 11%. This large gain confirms that recognizing discriminative channels for each category is crucial for fine-grained few shot classification. Furthermore, although the improvement of QAM is slightly lower than SAM, QAM is shown to be effective for all scenarios. We think that this is because the object-relevant channels not always represent discriminate channels. More importantly, the performances can be further boosted when combined. This validates that two sub-modules are complementary to one another.

5.2. Choice of Pooling Function

We also study the effects of pooling functions, as described in Tab. 8. As reported in the second and third rows, both pooling methods boost the performance since they are capable of representing the details of the objects. However, as shown in Fig. 2, max-pooling function has its limitation in that it is vulnerable to noise. So thus, we adopt the average-pooling function to predict discriminative channels.

5.3. Metric Compatibility

Following our baselines [7, 33, 34, 42], we employ the Euclidean distance for TDM. Since TDM is compatible with other metrics, we evaluate it with the cosine distance. As reported in Tab. 9, TDM improves the baseline. This validates that TDM's flexibility to commonly used distance metrics: the Euclidean and cosine distances.

6. Limitation

Since representing the whole objects could be rather harmful for classifying fine-grained categories unlike the general classification task, TDM is developed for highlighting features discriminative for fine-grained details. Thus, the benefits of TDM could be limited in coarse-grained tasks.

7. Conclusion

In this paper, we introduced Task Discrepancy Maximization (TDM), a tailored module for fine-grained few-shot classification. TDM produces channel weights that emphasize features of fine descriminative details to distinguish similar classes with two submodules: Support Attention Module (SAM) and Query Attention Module (QAM). Our extensive experiments on several fine-grained benchmarks validated the merits of our proposed TDM in terms of its effectiveness and high applicability with the prior few-shot classification methods. As a future direction, we will investigate how the significance for each channel vary in other computer vision tasks and extend this module to those tasks.

Acknowledgements. This work was supported in part by MCST/KOCCA (No. R2020070002), MSIT/IITP (No. 2020-0-00973, 2019-0-00421, 2020-0-01821, 2021-0-02003, and 2021-0-02068), and MSITKNPA/KIPoT (Police Lab 2.0, No. 210121M06).

References

[1] Arman Afrasiyabi, Jean-François Lalonde, and Christian Gagné. Associative alignment for few-shot image classification. In *European Conference on Computer Vision*, pages 18–35. Springer, 2020. 6

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *In International Conference on Learning Representations*, 2015. 3

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *In International Conference on Learning Representations*, 2019. 1, 6, 7

[4] Zhengyu Chen, Jixie Ge, Heshen Zhan, Siteng Huang, and Donglin Wang. Pareto self-supervised training for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13663–13672, 2021. 7

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[6] Yao Ding, Yanzhao Zhou, Yi Zhu, Qixiang Ye, and Jianbin Jiao. Selective sparse sampling for fine-grained image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6599–6608, 2019. 2, 5

[7] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. *In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2, 3, 5, 6, 7, 8

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017. 1, 2, 6

[9] Weifeng Ge, Xiangru Lin, and Yizhou Yu. Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3034–3043, 2019. 2, 5

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[11] Jie Hong, Pengfei Fang, Weihao Li, Tong Zhang, Christian Simon, Mehrtash Harandi, and Lars Petersson. Reinforced attention for few-shot learning and beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 913–923, 2021. 7

[12] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[13] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 124–140. Springer, 2020. 5

[14] Dahyun Kang, Heeseung Kwon, Juhong Min, and Minsu Cho. Relational embedding for few-shot classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8822–8833, 2021. 2, 3, 6

[15] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, volume 2. Citeseer, 2011. 6

[16] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6428, 2020. 1, 2

[17] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2019. 7

[18] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 6

[19] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019. 2

[20] Aoxue Li, Weiran Huang, Xu Lan, Jiashi Feng, Zhenguo Li, and Liwei Wang. Boosting few-shot learning with adaptive margin loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12576–12584, 2020. 2

[21] 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*, pages 7363–7372, 2021. 5

[22] Congcong Li, Dawei Du, Libo Zhang, Longyin Wen, Tiejian Luo, Yanjun Wu, and Pengfei Zhu. Spatial attention pyramid network for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 481–497. Springer, 2020. 2

[23] Junjie Li, Zilei Wang, and Xiaoming Hu. Learning intact features by erasing-inpainting for few-shot classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8401–8409, 2021. 5

[24] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7260–7268, 2019. 6

[25] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A

structured self-attentive sentence embedding. *In International Conference on Learning Representations*, 2017. 3

[26] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *European Conference on Computer Vision*, pages 438–455. Springer, 2020. 6

[27] Chuanbin Liu, Hongtao Xie, Zheng-Jun Zha, Lingfeng Ma, Lingyun Yu, and Yongdong Zhang. Filtration and distillation: Enhancing region attention for fine-grained visual categorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11555–11562, 2020. 2, 5

[28] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 6

[29] Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2218–2227, 2020. 6

[30] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012. 6

[31] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *In International Conference on Learning Representations*, 2017. 2

[32] Ryne Roady, Tyler L Hayes, Ronald Kemker, Ayesha Gonzales, and Christopher Kanan. Are open set classification methods effective on large-scale datasets? *Plos one*, 15(9):e0238302, 2020. 1, 3

[33] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4136–4145, 2020. 2, 3, 5, 6, 7, 8

[34] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 1, 2, 4, 5, 6, 7, 8

[35] 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*, pages 1199–1208, 2018. 1, 2, 6

[36] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 6

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3

[38] 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. 1, 2, 6

[39] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011. 6

[40] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019. 7

[41] Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6558–6567, 2019. 6

[42] Davis Wertheimer, Luming Tang, and Bharath Hariharan. Few-shot classification with feature map reconstruction networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8012–8021, 2021. 2, 3, 5, 6, 7, 8

[43] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2

[44] Chengming Xu, Yanwei Fu, Chen Liu, Chengjie Wang, Jilin Li, Feiyue Huang, Li Zhang, and Xiangyang Xue. Learning dynamic alignment via meta-filter for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5182–5191, 2021. 2, 3

[45] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8808–8817, 2020. 2, 3, 6, 7

[46] Baoquan Zhang, Xutao Li, Yunming Ye, Zhichao Huang, and Lisai Zhang. Prototype completion with primitive knowledge for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3754–3762, 2021. 1, 2, 3

[47] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12203–12213, 2020. 2, 6, 7

[48] Hongguang Zhang, Piotr Koniusz, Songlei Jian, Hongdong Li, and Philip HS Torr. Rethinking class relations: Absolute-relative supervised and unsupervised few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9432–9441, 2021. 7

[49] Jiabao Zhao, Yifan Yang, Xin Lin, Jing Yang, and Liang He. Looking wider for better adaptive representation in few-shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10981–10989, 2021. 7

[50] Heliang Zheng, Jianlong Fu, Zheng-Jun Zha, and Jiebo Luo. Looking for the devil in the details: Learning trilinear attention sampling network for fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5012–5021, 2019. 2, 5