arXiv:1912.08395v1 [cs.CV] 18 Dec 2019

Class Regularization: Improve Few-shot Image Classification by Reducing Meta Shift

†Da Chen, †Feng Mao, ‡Mingli Song, †Yuan He,
†Xiang Wu, §Jinqiao Wang, ¶Wenbin Li, ¶Yongliang Yang, †Hui Xue

†Alibaba Group
‡ Zhejiang University ¶University of Bath
§Institute of Automation, Chinese Academy of Sciences

***Abstract*—Few-shot image classification requires the classifier to robustly cope with unseen classes even if there are only a few samples for each class. Recent advances benefit from the meta-learning process where episodic tasks are formed to train a model that can adapt to class change. However, these tasks are independent to each other and existing works mainly rely on limited samples of individual support set in a single meta task. This strategy leads to severe meta shift issues across multiple tasks, meaning the learned prototypes or class descriptors are not stable as each task only involves their own support set. To avoid this problem, we propose a concise Class Regularization Network which aggregates the embedding features of all samples in the entire training set and further regularizes the generated class descriptor. The key is to train a class encoder and decoder structure that can encode the embedding sample features into a class domain with trained class basis, and generate a more stable and general class descriptor from the decoder. We evaluate our work by extensive comparisons with previous methods on two benchmark datasets (*MiniImageNet* and CUB). The results show that our method achieves state-of-the-art performance over previous work.**

I. Introduction

The human ability to understand new concepts with only a few examples has inspired the research on few-shot learning in the recent years. The main task is to achieve a learnt model that can classify new category given limited training data. Different from classification model [1], [2], [3] trained on large labeled dataset, few-shot learning model only relies on a few samples of each class (10, 5 ,or even less). This may easily lead to overfitting during training. To address this issue, Vinyals *et al.* [4] propose an attention mechanism which can learn an embedding of labelled samples from the support set and achieve good classification performance. This mechanism can be further enhanced by *episodes*, which aim to sub-sample categories and the associated data to simulate few-shot tasks during training. The episodic training process also benefits meta-learning [5], which can learn a probabilistic model to predict a decision boundary between categories given a few samples from each category.

By focusing on retrieving transferable embedding from the dataset, as well as the relation between images and the associated category descriptions, existing meta-learning approaches, such as ProtoNet [6] and RelationNet [7], prove to be effective for few-shot learning. However, they are often restricted by only targeting individual tasks during training, thus cannot efficiently explore the variation of all classes in the whole training set with a more general view. Due to the bias of chosen class subset for an individual training episode, the generated class descriptors/prototype for the same class in different episodes can be sparsely distributed in the feature space. As illustrated in Fig. 1A, the meta task target of ProtoNet [6], *i.e.,* class descriptors/prototypes (noted *descriptor* for the rest of the paper) of classes, are only required to be distinguishable under certain metric for the current task. The class descriptors in different tasks can still be distant to each other in the feature space, as individual tasks are considered separately. In this case, the class descriptors are not stable among tasks

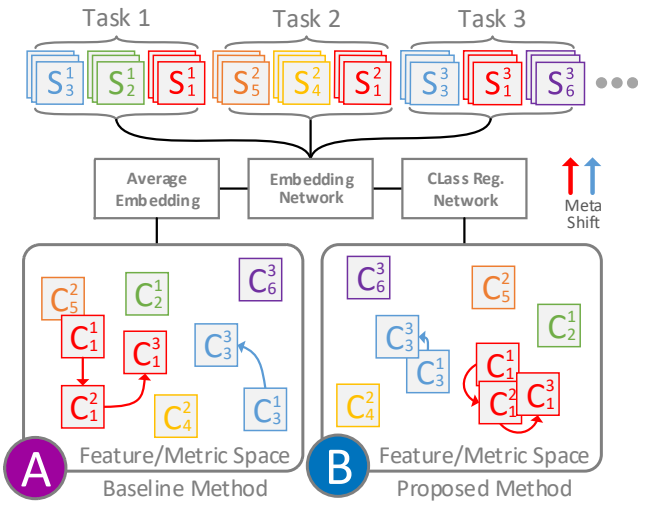

Fig. 1: A high-level description of meta shift. $\mathbf{S}_k^t$ and $\mathbf{C}_k^t$ are the support set and the class descriptor of the k-th class in task t respectively. During training, baseline method (A) generates class descriptor that is task-dependent and only concerns the classification result in the current task. The generated class descriptors across tasks are not stable and have shifting problem, *e.g.,* the first class is biased due to $\mathbf{C}_1^1$ and $\mathbf{C}_5^2$ are very close. The proposed method (B) utilizes a class domain with memory to regularize the class descriptor construction and avoid descriptor shifting.

during training, thus the average embedding easily causes misclassification in the test stage. This is defined as *meta shift* problem.

To solve the meta shift problem, we propose the *Class Regularization Network* to stabilize class descriptors across all tasks in the whole meta-learning process (see Fig. 1B). The resultant class descriptor can be generally applied to classify various query samples in a stable manner. The network first leverages comprehensive embedding sample features to form a class representation in the class domain composed by trained semantic basis via a class encoder. The generated class representation can then be decoded as a more stable and general class descriptor in the feature domain. The obtained class descriptors are applied to classify the query samples via a metric module which is combination of fix distance measure and a trainable relation module.

We demonstrate that the proposed method achieves state-of-the-art results on two benchmark datasets comparing to classic methods and more recent methods. For 1-shot, 5-shot and 10-shot tasks on *MiniImageNet*, it improves from 61.20% to 63.37%, 76.70% to 79.03%, and 80.80% to 83.30% respectively. For fine-grained dataset CUB, it achieves nearly 5% improvement on 1-shot task and improves 5-shot result from 81.90% to 83.53%. In addition, we illustrate that the proposed Class Regularization module effectively fine-tunes the embedding network to better tackle the covariate shift issue in few-shot learning. Further quantitative and qualitative evaluations show that the proposed method can effectively regularize the descriptor construction and improve classification results.

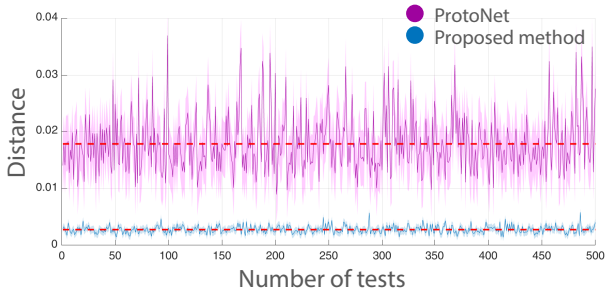

Fig. 2: Meta shift measurement: The statistics on the distance between class descriptors of class 'worm' and their mean in 500 tests for baseline method (A) and the proposed method (B). Five samples are randomly picked(5-shot task) to construct the class descriptor. This process is repeated for 500 tests to generate 500 class descriptors for each method. The Euclidean distance from the 'mean' class descriptor to the class descriptor in each test indicates the variation of the generated class descriptors which can be applied to measure the meta shift issue.

II. Related Work

Few-shot learning as an active research topic has been extensively studied. A number of works aim to improve the robustness of the training process for few-shot learning. Garcia *et al.* [8] propose a graph neural network according to the generic message-passing inference method. Zhao *et al.* [9] split the features to three orthogonal parts to improve the classification performance for few-shot learning, allowing simultaneous feature selection and dense estimation. Chen *et al.* [10] present a comprehensive analysis and investigate the cross-domain generalization ability for many existing few-shot learning methods. They also propose a new method which achieves state-of-the-art result on the CUB dataset [11]. Chen *et al.* [12] propose a Self-Jig algorithm to augment the input data in few-shot learning by synthesizing new images that are either labelled or unlabelled. Chu *et al.* [13] augment the input images by extracting varying sequences of patches on every forward-pass with discriminative observed information using maximum entropy reinforcement learning.

A popular strategy for few-shot learning is through meta-learning (also called learning-to-learn) with multi-auxiliary tasks [14], [15], [16], [17], [18], [19], [7], [4], [20]. The key insight is how to robustly accelerate the network learning progress without overfitting on limited training data. Finn *et al.* [15] propose MAML to search for the best initial weights through gradient descent for the network training, making the fine-tuning of the network easier. REPTILE [21] simplifies the complex computation of MAML by incorporating a L_2 loss, but still performs in high dimensional space. To reduce the complexity, Rusu *et al.* [22] propose a network called LEO to learn a low dimension latent embedding of the model. CAML [17] extends MAML by partitioning the model parameters into context parameters and shared parameters, enabling a larger network without over-fitting.

Another stream of meta-learning based approaches [23], [6], [7], [4] attempt to learn a deep embedding model that can effectively project the input samples to a specific feature space. Then the samples can be classified by the nearest neighbour criterion using a distance function such as Cosine distance or Euclidean distance, *etc.* Koch *et al.* [24] propose the Siamese network to extract embedding features from input images and converge images in the same class. Matching Network [4] utilizes an augmented neural network for feature embedding, forming the basis for metric learning. The most relevant work to ours is ProtoNet [6]. It proposes to model the class descriptor of each class with a simple average pooling on embedding sample features. However, the distance estimation only concerns the current task but not the whole train set. Many approaches extend ProtoNet by improving its class descriptor. RelationNet [7] exploits a relation network with a learnable non-linear comparator instead of a fixed linear one. TADAM [23] produces a task-dependent metric space based on conditioning a learner on the task set. Li *et al.* [25] propose to replace the distance measured at the image level to a local descriptor based image-to-class measure. Liu *et al.* [26] propose a transductive propagation network (TPN) which learns both the parameters of feature embedding and the graph construction. Unlike methods mentioned above which fine-tune the embedding network directly, Sun *et al.* [27]

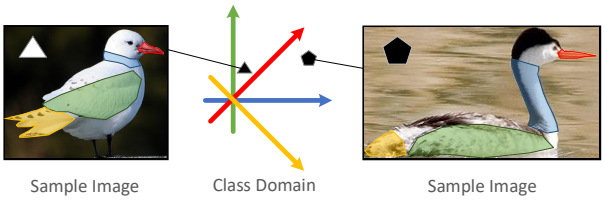

Fig. 3: Different components of bird are highlighted by different colors. This indicates the semantic basis in the class domain. The saliency of components implies the class location in class domain.

propose to apply a neuron-level scaling and shifting on the embedding network with a hard-task meta batch setting.

In this paper, we proposed a novel network that can learn much more stable class descriptors to generally represent all classes across the whole meta learning process. This avoids the meta shift and achieves state-of-the-art performance.

III. Class Regularization Network

In this section, we first present preliminaries in the meta-learning setting with independent episodic tasks and the subsequent meta shift problem, then give an overview of our approach, finally elaborate the details of our method.

A. Meta-learning for few-shot learning

Few-shot learning is a challenging problem as it is required to classify unseen queries with only limited data for training. One solution is to apply a meta-learning process composed by multiple K-way C-shot episodic classification tasks [4]. For each classification task, we have K classes with C samples from each class. The entire training dataset can be presented by $\mathcal{D} = \{(x_1, y_1), \ldots, (x_l, y_l)\}$, where l is the total number of samples in $\mathcal{D}$, and y is the corresponding label of sample x. For a specific task T, a support set and a query set are randomly selected from $\mathcal{D}$: (**a**) the support set for task T is denoted by $\mathcal{S} = \{(x_i, y_i)|i = 1, \ldots, m_s\}$, where $m_s = C \times K$ (K-way C-shot); (**b**) the query set is $\mathcal{Q} = \{(x_j, y_j)|j = 1, \ldots, m_q\}$, where m_q is the number of samples for query set in each meta task.

B. Meta shift problem

During the meta-learning process, most of the existing methods only focus on the selected samples in the support and query sets ($\mathcal{S}$ and $\mathcal{Q}$) of the current task T (such as 5-way, 5-shot classification), while ignoring the overall sample distribution within the training set $\mathcal{D}$. We spot that for different training tasks, support samples from the same class can be different as they are picked randomly from the entire training set $\mathcal{D}$. This leads to unstable episodic task training in meta learning. This unstable issue among meta learning tasks is defined as meta shift.

For example, as mention in Section II, the recent popular framework ProtoNet utilizes a feature embedding network to map all input samples in support set $\mathcal{S}_k (\mathcal{S}_k \subset \mathcal{S})$ from the k-th class to a mean vector c_k. They take this as a class descriptor in the common feature space:

$$c_k = \frac{1}{|\mathcal{S}_k|} \sum_{(x_i, y_i) \in \mathcal{S}_k} f_{\mathrm{EM}}(x_i), \tag{1}$$

where f_{EM} is the embedding function.

As it only focuses on the current task, the generated class descriptors are sparsely distributed in the feature space. As shown in Fig. 2, the variation of the generated class descriptors is considerably high (magenta curve), hindering the ability to accurately classify various query images.

Different from prior work, we aim at a better regularization over all the training tasks using a class regularization network. This network is supposed to resolve the meta shift problem and generate more stable feature descriptors (see Figure 1B) and further yield significantly improved performance for classification.

C. Key idea

To solve this problem, we consider how humans classify objects with few examples. Conceptually, humans can find key components of the objects and perform classification accordingly. For example, as shown in Fig. 3, the feather of wings (green), shape of neck (blue), mouth (red), and tail (yellow) can be very helpful to classify birds. These key components can be treated as semantic basis to form a class domain, and all classes can be represented in this domain according to the basis.

Inspired by the above, we propose a class encoder-decoder framework to form a class domain with trained semantic basis and then aggregate the sample features (within the same class) from the feature domain to the class domain. As there are much more samples of each class in the whole training set compared with individual support set, the embedding features are redundant to describe a class with sparse feature distribution. For each task, the aggregated sample features can update the encoder as a memory of how to form a better class representation in the class domain with semantic basis. Semantic basis based class representation can effectively avoid task-dependent problem which causes meta shift problem. This makes the encoder effective to describe various classes in a more stable way shown as the cyan curve in Fig. 2. Each class representation in the class domain is further reconstructed back into the feature domain to form a more stable and general class descriptor and classify query samples according to the metric module. Benefiting from a robust embedding network and a stable class descriptor, the proposed method improves classification performance. An overview of the proposed network is shown in Fig. 4.

D. Proposed method

In this subsection, we present the details of the network structure, including the embedding network, the class encoder-decoder network, and the metric module.

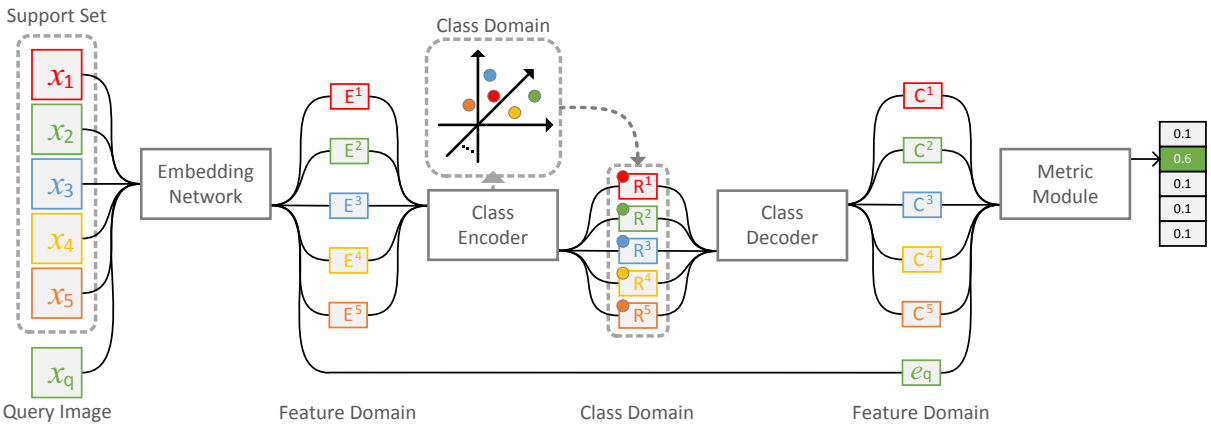

Fig. 4: Overview of the proposed Class Regularization Network. The Network is composed by four major parts: A feature **Embedding Network** (f_{EM}), **Class Encoder** (f_{EN}) for generating class representation with semantic basis in class domain, **Class Decoder** (f_{DE}) to transform the class representation back to feature domain and form a general class descriptor and **Metric Module** to compare class descriptor and query sample feature.

1) Embedding samples: Similar to previous metric learning based methods, the input samples are first fed into an embedding network.

$$e_i = f_{\mathrm{EM}}(x_i), \quad (x_i, y_i) \in \mathcal{S} \cup \mathcal{Q}, \tag{2}$$

where e_i is the embedding feature of the input sample (x_i, y_i) from either support set $\mathcal{S}$ or query set $\mathcal{Q}$.

2) Encoding classes: Given the embedding features, we use a class encoder f_{EN} to generate class representation with trainable semantic basis in the class domain. This is done by aggregating class features $\mathbf{E}^k$ composed by the embedding features for all samples in the k-th class in support set. The representation of the k-th class in the class domain is denoted as $\mathbf{R}^k$, which can be written as:

$$\mathbf{R}^k = f_{\mathrm{EN}}(\mathbf{E}^k). \tag{3}$$

For simplicity, we use $\mathbf{R}$ and $\mathbf{E}$ to replace $\mathbf{R}^k$ and $\mathbf{E}^k$ in the rest of this subsection.

Suppose class embedding $\mathbf{E}$ contains C sample features (C-shot task) and $\mathbf{R}$ is formed by N semantic basis, it is easy to see that $\mathbf{E}$ can be represented by a $C \times D$ matrix (D is the size of the embedding feature for a single sample), and $\mathbf{R}$ can be represented as a $N \times D$ matrix.

Similar to [28], by updating the form of semantic basis in each episodic task, we statistically gather information for class representation based on all samples in that class across all tasks. More specifically, the aggregation of the residual between each sample feature in one class and the semantic basis is applied to construct the class representation. For each class in an episodic task, the encoder constructs class representation based on multiple semantic basis. Mathematically, the representation of each semantic basis $\mathbf{R}_n$ (as the n-th row in $\mathbf{R}$) is updated based on $\mathbf{E}$ in the following way:

$$\mathbf{R}_n = \sum_{i=1}^{C} a_n(\mathbf{E}_i)(\mathbf{E}_i - \mathbf{R}_n), n = [0, ..., N-1] \tag{4}$$

where $\mathbf{E}_i$ is the i-th sample feature in the current task. $a_n(\mathbf{E}_i) = 1$ if $\mathbf{R}_n$ is the closest to $\mathbf{E}_i$, otherwise $a_n(\mathbf{E}_i) = 0$. The sum of residuals $(\mathbf{E}_i - \mathbf{R}_n)$ are further assigned to $\mathbf{R}_n$. The weighted residual indicates the saliency of the sample feature response to each semantic basis. After a normalization operation, the output will be a $N \cdot D \cdot 1$ class representation in the class domain.

However, the operation based on Eq. 4 is not differentiable due to the discontinuities when assigning $a_n(\mathbf{E}_i)$ to semantic basis $\mathbf{R}_n$. To solve this problem, $a_n(\mathbf{E}_i)$ is replaced by a soft assignment of sample features to the semantic basis:

$$\bar{a}_n(\mathbf{E}_i) = \frac{e^{-\alpha \|\mathbf{E}_i - \mathbf{R}_n\|^2}}{\sum_{n'} e^{-\alpha \|\mathbf{E}_i - \mathbf{R}_{n'}\|^2}}, \tag{5}$$

where parameter α controls the sensitivity to the residual.

It can be seen that the sample feature $\mathbf{E}_i$ is assigned to basis $\mathbf{R}_n$ in a weighted manner using weight $\bar{a}_n(\mathbf{E}_i)$. And the class representation $\mathbf{R} \in \mathbb{R}^{N \times D}$ class domain. The encoder parameter θ_{EN} can be represented by $\theta_{\mathrm{EN}} = [\{2\alpha\mathbf{R}_n\}, \{-\alpha \|\mathbf{R}_n\|^2\}, \{\mathbf{R}_n\}]$. More details can be found in the supplementary material.

3) Decoding classes: With the aforementioned encoded class representation $\mathbf{R}$ in the class domain, we are able to decode and transform them from class domain back into feature domain as a class descriptor which is more general and stable.

In the class decoder, we include a transformation function f_{DE} to obtain the general class descriptor $\mathbf{C}$ given the class representation $\mathbf{R}$ in the class domain:

$$\mathbf{C} = f_{\mathrm{DE}}(\mathbf{R}). \tag{6}$$

Note that the dimension of the transformed class descriptor $\mathbf{C}$ is also D, which is the same as the embedding feature as they are both in the feature domain.

As shown in Fig. 1 and Fig. 2, ProtoNet simply applies average pooling on sample features of one class to obtain class descriptor. As the input support set only has a small number of samples in each task, the associated features are not consistent among tasks, raising meta shift issue where the same class has unstable descriptors for different tasks.

On the other hand, our method extracts better class descriptor based on the distribution of all classes. This is done by decomposing the class representation into N trainable semantic basis in a class domain. The decoded class descriptor proves to be more stable among all tasks. More details are discussed in Section IV.

4) Metric Module: To classify the query image, classic few-shot classification methods such as ProtoNet [6] apply a simple fixed distance metric *i.e.,* Cosine distance, Euclidean distance, *etc.* In this work, we include a general relation module [7] along with a fixed distance metric via a co-training mechanism as a metric module, resulting in a more robust similarity measurement for comparing the embedding feature $f_{\mathrm{EM}}(q)$ of a query image q to the general class descriptor $\mathbf{C}_k$ of the k-th class. Here we discuss the metric module in details.

Similar to ProtoNet [6], we employ a Euclidean distance function and produce a distribution over all classes given a sample q from the query set $\mathcal{Q}$. The distribution is based on a softmax over the distance between the sample feature (in the query set) and the general class descriptor. The loss function can be defined as:

$$L_E = d(f_{\mathrm{EM}}(q), \mathbf{C}_k) + \log \sum_{k'} \exp(-d(f_{\mathrm{EM}}(q), \mathbf{C}_{k'})) \quad (7)$$

Apart from the fixed distance function, we also include a trainable relation module. Given the embedding feature of the query image $f_{\mathrm{EM}}(q)$ and class descriptor $\mathbf{C}_k$, the output of the relation module indicates the correlation between the two. We treat this output as the relation score [7] among k classes and all query images. Similar to Eq. 7, given the correct class label y_q for q, the loss for the relation module can be written as:

$$L_R = -f_R(\varsigma(f_\phi(q), \mathbf{C}_k)) + \log \sum_{k'} \exp(f_R(\varsigma(f_\phi(q), \mathbf{C}_{k'}))), \quad (8)$$

where f_R is the relation function, ς denotes concatenation. The total loss of the network can then be summarized as:

$$L = \alpha_1 * L_E + \alpha_2 * L_R + \alpha_3 * \|\Theta\|^2, \quad (9)$$

where $\alpha_1, \alpha_2, \alpha_3$ are the weights of the two losses and the regularization term, $\Theta = [\theta_{\mathrm{EN}}, \theta_{\mathrm{DE}}, \theta_{\mathrm{RM}}]$ are the training parameters of class encoder, class decoder and relation module. The algorithmic process of this module and the entire network can be found in the supplementary material.

IV. Evaluation

In this section, we first describe the network architecture and the datasets in our evaluation, then detail the training procedure, finally show quantitative comparisons with baseline methods, and a thorough discussion of the advantage over the most relevant work [6].

A. Network architecture

The overall architecture of the proposed network is shown in Fig. 4. Noted that it can incorporate different feature embedding networks. In our experiments, we apply two networks that are usually used in the few-shot image classification field [25], [27], including ResNet12 which is a residual network [1] with three residual blocks followed by an average pooling layer for general feature embedding, and 4 Conv with four convolutional blocks for feature extraction. The class encoder has a convolutional block and a softmax operator for soft-assignment (see details in the supplemental) to produce a sparse matrix which includes all weights for semantic basis. With a multiplication between the weights and the residuals followed by a normalization layer, the class representation composed by semantic basis in the class domain is obtained. The decoder exploits a normal convolutional block to transform the class representation from the class domain to general class descriptor in the feature domain. To compare the obtained general class descriptor with the embedding features of the samples in the query set, a fixed distance metric (*i.e.,* Euclidean distance) and a learnable relation module (two Conv + two FC) based similarity measurement are jointly applied, while a cross-entropy loss is employed as classification loss. The network then updates the metric module along with the encoder-decoder structure, and fine-tune the feature embedding network simultaneously.

B. Dataset

MiniImageNet dataset, as proposed in [4], is a benchmark to evaluate few-shot learning methods. This dataset is a randomly selected subset from *ImageNet* – a popular image classification dataset. *MiniImageNet* contains 60,000 images from only 100 classes, and each class has 600 images. We also follow the data split strategy in [5] to sample images of 64 classes for training, 16 classes for validation, 20 classes for test.

Caltech-UCSD CUB-200-2011 dataset [11] is a dataset for fine-grained classification. The CUB-200-2011 dataset contains 200 classes of birds with 11788 images in total. For few-shot learning classification task, we follow the split in [29] for evaluation. 200 species of birds are randomly split to 100 classes for training, 50 classes for validation, and 50 classes for test.

Embedding Net	Image Size	Basis	l-rate	Pre l-rate	$[\alpha_1, \alpha_2, \alpha_3]$
ResNet12	$\{80 \times 80\}$	16	$1e^{-3}$	$1e^{-4}$	$[1/8, 1, 1]$
Conv	$\{84 \times 84\}$	8	$1e^{-3}$	$1e^{-4}$	$[1/2, 1, 1]$

TABLE I: Training parameters for different embedding networks on *MiniImageNet* and CUB datasets.

Baselines	Embedding Net	1-Shot 5-Way	5-Shot 5-Way	10-Shot 5-Way
Matching Networks [4]	4 Conv	43.56 ± 0.84%	55.31 ± 0.73%	-
MAML [15]	4 Conv	48.70 ± 1.84%	63.11 ± 0.92%	-
RelationNet [7]	4 Conv	50.44 ± 0.82%	65.32 ± 0.70%	-
REPTILE [21]	4 Conv	49.97 ± 0.32%	65.99 ± 0.58%	-
ProtoNet [6]	4 Conv	49.42 ± 0.78%	68.20 ± 0.66%	74.30%
Baseline* [10]	4 Conv	41.08 ± 0.70%	54.50% ± 0.66	-
Spot&learn [13]	4 Conv	51.03 ± 0.78%	67.96% ± 0.71	-
DN4 [25]	4 Conv	51.24 ± 0.74%	**71.02% ± 0.64**	-
Ours_Conv	4 Conv	**52.0 ± 0.20%**	70.98% ± 0.16	-
Discriminative k-shot [30]	ResNet34	56.30 ± 0.40%	73.90 ± 0.30%	78.50 ± 0.30%
Self-Jig(SVM) [12]	ResNet50	58.80 ± 1.36%	76.71 ± 0.72%	-
Qiao-WRN [19]	Wide-ResNet28	59.60 ± 0.41%	73.74 ± 0.19%	-
LEO [22]	Wide-ResNet28	61.76 ± 0.08%	77.59 ± 0.12%	-
SNAIL [18]	ResNet12	55.71 ± 0.99%	68.88 ± 0.92%	-
ProtoNet$^+$ [6]	ResNet12	56.50 ± 0.40%	74.2 ± 0.20%	78.60 ± 0.40%
RelationNet$^+$ [7]	ResNet12	58.20 ± 0.30%	74.35 ± 0.23%	78.65 ± 0.30%
CAML [17]	ResNet12	59.23 ± 0.99%	72.35 ± 0.71%	-
TPN [26]	ResNet12	59.46%	75.65%	-
MTL [27]	ResNet12	61.20 ± 1.8%	75.50 ± 0.8%	-
DN4 [25]	ResNet12	54.37 ± 0.36%	74.44 ± 0.29%	-
TADAM [23]	ResNet12	58.50%	76.70%	80.80%
Ours_Res	ResNet12	**63.37 ± 0.14%**	**79.03 ± 0.16%**	**83.30 ± 0.13%**

TABLE II: Few-shot classification accuracy results on *MiniImageNet* on 1-shot 5-way, 5-shot 5-way and 10-shot 5-way tasks. All accuracy results are reported with 95% confidence intervals. For each task, the best-performing method is highlighted. '-': the results are not reported. '+': re-implementation results for a fair comparison.

Baselines	Embedding Net	1-Shot 5-Way	5-Shot 5-Way
MatchingNet [4]	4 Conv	61.16 ± 0.89	72.86 ± 0.70
MAML [15]	4 Conv	55.92 ± 0.95%	72.09 ± 0.76%
ProtoNet [6]	4 Conv	51.31 ± 0.91%	70.77 ± 0.69%
MACO [29]	4 Conv	60.76%	74.96%
RelationNet [7]	4 Conv	62.45 ± 0.98%	76.11 ± 0.69%
ProtoNet$^+$ [6]	4 Conv	63.52 ± 0.25%	79.06 ± 0.20%
Baseline++ [10]	4 Conv	60.53 ± 0.83%	79.34 ± 0.61%
DN4-DA [25]	4 Conv	53.15 ± 0.84%	81.90 ± 0.60%
Ours_conv	4 Conv	**67.85 ± 0.24%**	**83.53 ± 0.16%**

TABLE III: Few-shot classification accuracy results on CUB dataset [11] on 1-shot 5-way task, 5-shot 5-way task. All accuracy results are reported with 95% confidence intervals. For each task, the best-performing method is highlighted.

C. Training details

Several recent works show that a typical training process can include a pre-trained network [19], [22] or employ co-training [23] for feature embedding. This can significantly improve the classification accuracy. In this paper, we adapt the training strategy from [19] to pre-train the feature embedding network as discussed in Section IV-A. In our training process, we follow the standard setup as applied by most few-shot learning frameworks, introduced in Section III-A, to train the embedding network, class encoder and decoder along with the metric module. Stochastic gradient descent (SGD) is used as the optimizer for ResNet embedding network, while ADAM is chosen as the optimizer for Conv embedding network. During testing, we follow the setup in [6]. 15 query images per class are batched in the testing episode. The accuracy is obtained by averaging 600 test tasks which generate test batches with random classes and random samples. Table I shows the training parameters with different embedding networks on the two datasets. The training time for proposed method is closed to ProtoNet as the class regularization module has a simple architecture(75 second per epoch on a single P100 machine for both methods). The number of semantic basis is an important hyper-parameter. See supplemental for its relation with classification accuracy.

D. Comparing with other baselines

As detailed in Table II, the proposed method is compared with baselines including classic methods and more recent methods. It can be seen that our method achieves state-of-the-art results in all three tasks with different embedding networks on *MiniImageNet* dataset. For 1-shot 5-way test, based on ResNet12 embedding network, our approach achieves 6.87% and 2.17% improvement over ProtoNet$^+$ and MTL [27] respectively. The former is an amended variant of ProtoNet [6] using pre-trained network as embedding network for a fair comparison. The later is a more recent work. For 5-shot 5-way test, we observe a similar accuracy improvement of 4.83% over ProtoNet$^+$. Note that our method also yields a margin of increment over the current state-of-the-art approach LEO [22], which applies a 28-layer wide residual network. Furthermore, we see that the performance of the proposed method is better while receiving more images(shots) as input. For example, our accuracy is 79.03% in 5-shot 5-way test against 63.37% for 1-shot 5-way. This is because more samples per-class provide more information to our network, thus further improve the descriptiveness of the learned aggregated class descriptor. Moreover, as shown in Table II, most of the existing methods apply a 12-layer residual network [1] or a 4 Conv blocks network for feature embedding. Some of them

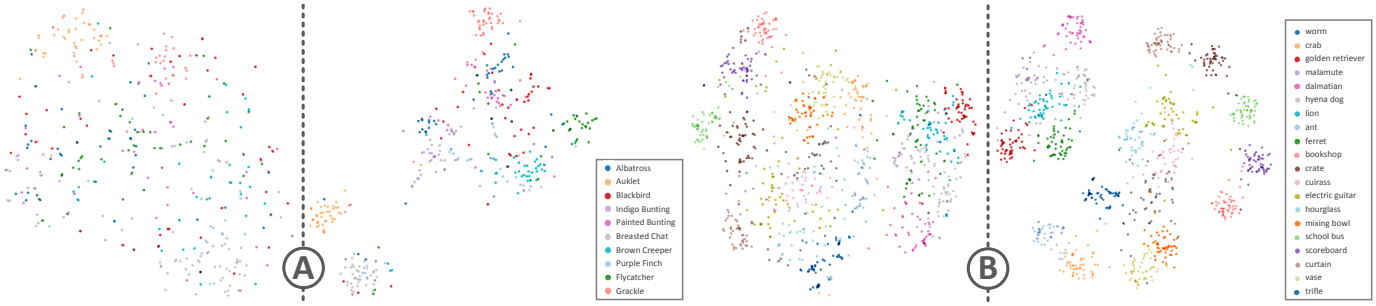

Fig. 5: t-SNE plot of latent space features for 20 class in test set in *MiniImageNet* and 10 classes in CUB datasets. 50 samples are randomly selected from each classes. **(A)** ProtoNet$^+$ (Left) and Ours (Right) in the CUB dataset; **(B)** same as (A) but in *MiniImageNet* dataset. Colors of the feature points represent class labels as in the right box.

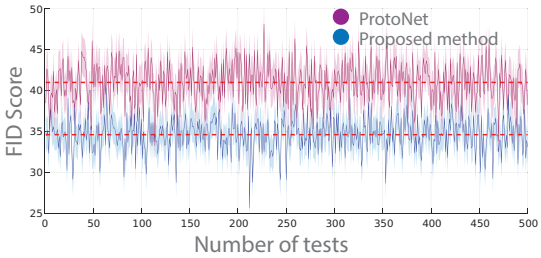

Fig. 6: FID score [31]: The curves show the result of 500 tests. For each test, 25 samples(5-way 5-shot) for training set and 75 samples(15 queries/class) for query set are randomly chosen to obtain its sample features by both methods. FID score measures the similarity of training and testing feature distribution for baselines.

even apply a deeper feature embedding network with much more parameters to train, such as ResNet-50 used by Sel-Jig [12], ResNet-34 used by Discriminative k-shot [30], and a 28-layer wide residual network used by both LEO [22] and Qiao-WRN [19]. Even so, our method still achieves better performance applying a simpler embedding network with 30% less parameters comparing to state-of-the-art method LEO.

Table III summarizes the comparisons on the CUB dataset. Our method yields the highest accuracy from all trials. In the 1-shot 5-way test, we achieves 67.85% accuracy which improves to the classic Matching Networks [4] by 6.69%. More significant improvement is achieved for 5-shot 5-way test. Our classification accuracy is 83.53%, an improvement of 4.47% over ProtoNet$^+$. Comparing to the recently proposed Baseline++ [10], our method demonstrates a significant improvement of 7.32% and 4.19% in tests.

Note that different baseline methods take input images with different sizes. Although most methods set image size to $\{84\times 84\}$, larger image can help to improve classification accuracy with better data augmentation condition. For instance, Sel-Jig [12] relies on 224×224 images with data augmentation, while our method only requires 80×80 images without data augmentation.

E. Effectiveness of Class Regularization

Meta Shift: To highlight the advantage of the proposed method against other non-class-regularization baseline (*i.e.,* ProtoNet [6]), we measure the stability of the class descriptors of the proposed method against other methods having meta shift issue. Note that for a fair comparison, we use adapted ProtoNet$^+$ with the same pre-trained embedding network. For each class in the test set of *MiniImageNet*, we randomly pick 5 samples to construct the class descriptor. We repeat this process for 500 tests to generate 500 class descriptors for each baseline. The Euclidean distance from the 'mean' class descriptor to specific class descriptor in each test indicates the variation of the generated class descriptors. Fig. 2 shows statistics for the class 'worm'. The proposed method (cyan curve with much less variation) is able to generate more stable class descriptors comparing to ProtoNet$^+$ (magenta curve with severe feature shifting). This results in better performance for the few-shot classification task. More comparisons of meta shift express in the supplementary material.

Covariate Shift Given limited training data (even one sample per class for 1-shot task), when dealing with unseen classes during meta testing, the training data distribution is likely to be different from the distribution of the query data (covariate shift [32]). It requires a generalized embedding network to embed them into a similar feature distribution and be classified with a stable class descriptor from training.

Similar to the qualitative evaluation in [22], we use t-SNE projection to visualize the feature space of ProtoNet$^+$ and our method. For clarity, we randomly choose 50 samples from each class in the test set and extract their features from the embedding network. The t-SNE plot of the feature space is shown in Fig 5 (the plot of all samples is included in the supplementary material). The feature space of the proposed method is observed to be different from the one of ProtoNet$^+$ in both *MiniImageNet* and CUB dataset. Our embedding features are densely clustered within each class while sparsely distributed among different classes, demonstrating intra-class commonality and inter-class distinctiveness. As training and testing sets are randomly selected from these samples, this qualitative result proves that the proposed class regularization effectively fine-tunes the embedding network and better deal

Dataset	Framework	1-shot	5-shot
MiniImageNet	Res+Euc	56.50 ± 0.40	74.2 ± 0.20
	Res+ACD+Euc	62.97 ± 0.16%	78.87 ± 0.15%
	Res+ACD+(Euc+R)	**63.37 ± 0.14%**	**79.03 ± 0.16%**
CUB	Conv+Euc	63.52 ± 0.25%	79.06 ± 0.20%
	Conv+ACD+Euc	66.80 ± 0.22%	82.0 ± 0.15%
	Conv+ACD+(Euc+R)	**67.85 ± 0.24%**	**83.53 ± 0.16%**

TABLE IV: Ablation study on two datasets by comparing ProtoNet$^+$ (top), our method without relation module (middle), and with relation module (bottom).

with covariate shift issues.

We also compare the similarity of training sample features against query sample features. This quantitatively measures the efficiency of the embedding networks which are fine-tuned by each baseline. In this work, we apply Fréchet Inception Distance(FID) score [31], which is widely used in generative model evaluation, to measure the feature distribution difference between the training samples and testing samples. Ideally, with a close distribution, the classifier can perform better. As shown in Fig 6, the embedding network fine-tuned by the proposed class regularization method consistently less than the embedding network fine-tuned by ProtoNet, indicating higher similarity and a better embedding network.

The qualitative and quantitative results show that the proposed method can not only tackle the meta shift issue but also efficiently fine-tune the embedding network to alleviate the covariate shift issue in few-shot learning. These two benefits make the proposed method significantly outperform other baselines.

Metric module In addition, we further perform an ablation study by simply applying Euclidean distance for image classification (same as ProtoNet$^+$). The result is shown in Table IV. For clarity, we also show the performance of ProtoNet$^+$ with the same embedding network. It can be seen that due to the effectiveness of the proposed class regularization method, the network without relation module still outperforms baselines listed in Table II and Table III. Relation module combined with Euclidean distance further improves the performance.

V. Conclusion and Future Work

In this paper, to tackle the meta shift problem, we propose a novel meta-learning framework that employs class regularization for better feature embedding and better class descriptor over the entire training set. We utilize an aggregation of image embeddings for better extraction of finer details during the training process, along with an effective representation of the target classes using aggregated descriptors learned from the framework. We evaluate the proposed framework by comparing with existing few-shot learning methods on the *MiniImageNet* and CUB datasets. The state-of-the-art performance demonstrates the advantage of our framework over previous works.

Several directions can be explored in the future. One way to improve the effectiveness of the class regularization module would be to involve task condition/embedding along with the current image embedding. Another direction is to further strengthen the embedding network by using FPN [33] or attention based techniques [34].

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 1, 5, 6

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 1

[3] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, vol. 4, 2017, p. 12. 1

[4] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *Advances in neural information processing systems*, 2016, pp. 3630–3638. 1, 2, 3, 5, 6, 7

[5] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International Conference on Learning Representations*, 2017. 1, 5

[6] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4077–4087. 1, 2, 5, 6, 7, 11

[7] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1199–1208. 1, 2, 5, 6, 12

[8] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," *arXiv preprint arXiv:1711.04043*, 2017. 2

[9] B. Zhao, X. Sun, Y. Fu, Y. Yao, and Y. Wang, "Msplit lbi: Realizing feature selection and dense estimation simultaneously in few-shot and zero-shot learning," *arXiv preprint arXiv:1806.04360*, 2018. 2

[10] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HkxLXnAcFQ 2, 6, 7

[11] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011. 2, 5, 6

[12] C. Zitian, F. Yanwei, C. Kaiyu, and J. Yu-Gang, "Image block augmentation for one-shot learning." in *AAAI*, 2019. 2, 6, 7

[13] W.-H. Chu, Y.-J. Li, J.-C. Chang, and Y.-C. F. Wang, "Spot and learn: A maximum-entropy patch sampler for few-shot image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6251–6260. 2, 6

[14] M. Douze, A. Szlam, B. Hariharan, and H. Jégou, "Low-shot learning with large-scale diffusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3349–3358. 2

[15] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1126–1135. 2, 6

[16] B. Hariharan and R. Girshick, "Low-shot visual recognition by shrinking and hallucinating features," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3018–3027. 2

[17] X. Jiang, M. Havaei, F. Varno, G. Chartrand, N. Chapados, and S. Matwin, "Learning to learn with conditional class dependencies," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=BJfOXnActQ 2, 6

[18] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141*, 2017. 2, 6

[19] S. Qiao, C. Liu, W. Shen, and A. L. Yuille, "Few-shot image recognition by predicting parameters from activations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7229–7238. 2, 6, 7

[20] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7278–7286. 2

[21] A. Nichol and J. Schulman, "Reptile: a scalable metalearning algorithm," *arXiv preprint arXiv:1803.02999*, vol. 2, 2018. 2, 6

[22] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=BJgklhAcK7 2, 6, 7

[23] B. Oreshkin, P. R. López, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 719–729. 2, 6

[24] G. Ãğ, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, 2015. 2

[25] W. Li, L. Wang, J. Xu, J. Huo, Y. Gao, and J. Luo, "Revisiting local descriptor based image-to-class measure for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7260–7268. 2, 5, 6

[26] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. Hwang, and Y. Yang, "LEARNING TO PROPAGATE LABELS: TRANSDUCTIVE PROPAGATION NETWORK FOR FEW-SHOT LEARNING," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SyVuRiC5K7 2, 6

[27] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 403–412. 2, 5, 6

[28] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *European conference on computer vision*. Springer, 2014, pp. 392–407. 4

[29] N. Hilliard, L. Phillips, S. Howland, A. Yankov, C. D. Corley, and N. O. Hodas, "Few-shot learning with metric-agnostic conditional embeddings," *arXiv preprint arXiv:1802.04376*, 2018. 5, 6

[30] M. Bauer, M. Rojas-Carulla, J. B. Świątkowski, B. Schölkopf, and R. E. Turner, "Discriminative k-shot learning using probabilistic models," *arXiv preprint arXiv:1706.00326*, 2017. 6, 7

[31] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637. 7, 8

[32] M. Sugiyama, M. Krauledat, and K.-R. MÃžller, "Covariate shift adaptation by importance weighted cross validation," *Journal of Machine Learning Research*, vol. 8, no. May, pp. 985–1005, 2007. 7

[33] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection." in *CVPR*, vol. 1, no. 2, 2017, p. 3. 8

[34] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," *arXiv preprint arXiv:1704.06904*, 2017. 8

Supplemental Materials

VI. Introduction

In this supplementary document, we present 1) additional results to show the effectiveness of the proposed class regularization network; 2) mathematical details of the class encoder-decoder module and metric module; 3) the algorithmic procedure of the proposed method.

VII. Effectiveness of Class Regularization on Meta shift reduction: More qualitative results and meta shift plots

As mentioned in the main paper, we also plot the t-SNE visualization of the feature space for ProtoNet$^+$ and the proposed method. Fig. 7 includes all the samples of the test sets for both MiniImageNet and CUB datasets.

As an important component of the proposed framework, the number of the semantic basis in the class domain is essential to the performance of the proposed method. Fig. 8 shows the effectiveness of the semantic basis number on the accuracy of the proposed method.

Fig. 9 shows additional meta shift plots of more classes similar to the plot in Fig. 2 in the main paper. Note that all classes are from the test set of MiniImageNet ('worm', 'crab', 'golden retriever', 'malamute', 'dalmatian', 'hyena dog', 'lion', 'ant', 'ferret', 'bookshop').

VIII. Detailed math of class domain construction and metric module

A. Class domain construction

In this section, we include the background mathematical details of the proposed method. The notation is consistent with Section 3 in the main paper.

The embedding function of the proposed method can provide a feature embedding of given samples:

$$\begin{aligned} \mathbf{E}_k &= [e_1, \ldots, e_C]^T \\ \mathbf{E}_i &= f_{\mathbf{EM}}(x_i) \quad (x_i, y_i) \in \mathcal{S}, \end{aligned} \tag{10}$$

where $\mathbf{E}_k$ is a $C \times D$ tensor that represents the embedding features for all C samples from the k-th class in support set $\mathcal{S}$ for the current training task. D is the size of the embedding features for each input sample. The class representation in the class domain can be written as:

$$\mathbf{R}_k = f_{\mathbf{EN}}(\mathbf{E}_k), \tag{11}$$

where $f_{\mathbf{EN}}$ is the class encoder function. We compose the class representation with the semantic basis which can be treated as a weighted summation of residuals:

$$\mathbf{R}(j,n) = \sum_{i=1}^{C} a_n(\mathbf{E}_i)(\mathbf{E}_i(j) - \mathbf{R}_n(j)), \tag{12}$$

where $\mathbf{E}_i(j)$ is the j-th dimension of the i-th embedding feature, $\mathbf{R}_n(j)$ is the j-th dimension of the representation on n-th semantic basis. $a_n(\mathbf{E}_i) = 1$ if representation on semantic basis $\mathbf{R}_n$ is the closest to embedding feat $\mathbf{E}_i$, otherwise $a_n(\mathbf{E}_i) = 0$.

$$\bar{a}_n(\mathbf{E}_i) = \frac{e^{-\alpha\|\mathbf{E}_i - \mathbf{R}_n\|^2}}{\sum_{n'} e^{-\alpha\|\mathbf{E}_i - \mathbf{R}_{n'}\|^2}}, \tag{13}$$

where α is a parameter that controls the sensitivity to the residual.

In this supplementary material, we expand Eq. 13 and obtain the final representation of Eq. 16. Eq 13 can be expanded as:

$$\begin{aligned} \bar{a}_n(\mathbf{E}_i) &= \frac{e^{-\alpha\left\|\mathbf{E}_i^2 - 2\cdot\mathbf{E}_i\cdot\mathbf{R}_n + \mathbf{R}_n^2\right\|}}{\sum_{n'} e^{-\alpha\left\|\mathbf{E}_i^2 - 2\cdot\mathbf{E}_i\cdot\mathbf{R}_{n'} + \mathbf{R}_{n'}^2\right\|}} \\ &= \frac{e^{-\alpha(\|\mathbf{E}_i\|^2 - 2\cdot\mathbf{E}_i\cdot\mathbf{R}_n + \|\mathbf{R}_n\|^2)}}{\sum_{n'} e^{-\alpha(\|\mathbf{E}_i\|^2 - 2\cdot\mathbf{E}_i\cdot\mathbf{R}_{n'} + \|\mathbf{R}_{n'}\|^2)}} \\ &= \frac{e^{-\alpha\|\mathbf{E}_i\|^2} \cdot e^{2\alpha\mathbf{E}_i\cdot\mathbf{R}_n} \cdot e^{-\alpha\|\mathbf{R}_n\|^2}}{\sum_{n'} e^{-\alpha\|\mathbf{E}_i\|^2} \cdot e^{2\alpha\mathbf{E}_i\cdot\mathbf{R}_{n'}} \cdot e^{-\alpha\|\mathbf{R}_{n'}\|^2}}, \end{aligned} \tag{14}$$

as the summation in denominator is only related to n' representation on semantic basis. Eq. 14 can be transformed as:

$$\begin{aligned} \bar{a}_n(\mathbf{E}_i) &= \frac{e^{-\alpha\|\mathbf{E}_i\|^2} \cdot e^{2\alpha\mathbf{E}_i\cdot\mathbf{R}_n} \cdot e^{-\alpha\|\mathbf{R}_n\|^2}}{e^{-\alpha\|\mathbf{E}_i\|^2} \sum_{n'} e^{2\alpha\mathbf{E}_i\cdot\mathbf{R}_{n'}} \cdot e^{-\alpha\|\mathbf{R}_{n'}\|^2}} \\ &= \frac{e^{2\alpha\mathbf{E}_i\cdot\mathbf{R}_n} \cdot e^{-\alpha\|\mathbf{R}_n\|^2}}{\sum_{n'} e^{2\alpha\mathbf{E}_i\cdot\mathbf{R}_{n'}} \cdot e^{-\alpha\|\mathbf{R}_{n'}\|^2}}, \end{aligned} \tag{15}$$

It can be seen that the embedding feature $\mathbf{E}_i$ is assigned to semantic basis $\mathbf{R}_n$ using Eq. 12 in a weighted manner, where the weight $\bar{a}_n(\mathbf{E}_i)$ is noted in Eq. 13. And Eq. 12 can be re-written as follows:

$$\mathbf{R}(j,n) = \sum_{i=1}^{C} \frac{e^{-\alpha\|\mathbf{E}_i - \mathbf{R}_n\|^2}}{\sum_{n'} e^{-\alpha\|\mathbf{E}_i - \mathbf{R}_{n'}\|^2}} (\mathbf{E}_i(j) - \mathbf{R}_n(j)) \tag{16}$$

Based on Eq. 12 and Eq. 15, Eq. 16 can be rewritten as:

$$\mathbf{R}(j,n) = \sum_{i=1}^{C} \frac{e^{2\alpha\mathbf{E}_i\cdot\mathbf{R}_n} \cdot e^{-\alpha\|\mathbf{R}_n\|^2}}{\sum_{n'} e^{2\alpha\mathbf{E}_i\cdot\mathbf{R}_{n'}} \cdot e^{-\alpha\|\mathbf{R}_{n'}\|^2}} (\mathbf{E}_i(j) - \mathbf{R}_n(j)) \tag{17}$$

Based on Eq. 17, given embedding features E of samples from a class, we can get the class representation $\mathbf{R} \in \mathbb{R}^{N\times D}$ with N semantic basis in the class domain. The parameter θ_{EN} of encoder function $f_{\mathrm{EN}}(\mathbf{E}_k)$ can be represented as:

$$\theta_{\mathrm{EN}}(n) = [\{2\alpha\mathbf{R}_n\}, \{-\alpha\,\|\mathbf{R}_n\|^2\}, \{\mathbf{R}_n\}]. \tag{18}$$

The obtained $\theta_{\mathrm{EN}}(n)$ set is the trainable parameter set of the encoder for semantic basis $\mathbf{R}_n$.

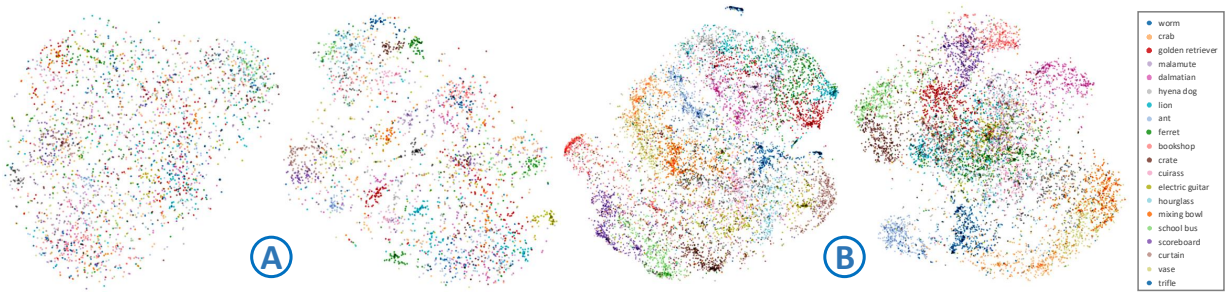

Fig. 7: t-SNE plot of feature spaces (with all samples) of amended ProtoNet (ProtoNet-A/ProtoNet*) and the proposed Network (Ours_res/Ours_conv). **(A)**: Embedding feature space of Amended ProtoNet (Left) and Ours (Right) in the CUB dataset. **(B)**: Same as **(A)** but in MiniImageNet dataset.

Algorithm 1 Algorithmic process for Class Regularization Network. $\mathcal{D}$ is the entire training set; $\mathcal{N}$ is the total number of class in $\mathcal{D}$; N_e is the number of episodes for meta training; N_t is the number of tasks for one meta episode; V is the classes selected for support set; K is the number of classes sampled for each task in one meta episode; $\mathcal{D}_k$ is the image set of the k-th class in $\mathcal{D}$; $\mathcal{S}_k$ is the sampled support image set of the k-th class, and each image is labelled; $\mathcal{Q}_k$ is the query image set to test the re-construction of the k-th class, and each image is unlabelled; $N_{\mathcal{S}}$ is the size of each support image set; m_q is the size of each test/query image set;

Initialize Proposed Network
for each $i_e \in [1, N_e]$ **do** ▷ meta training episodes e
  for each $i_t \in [1, N_t]$ **do** ▷ tasks in one episode t
    V_t = RandomSample($[1, \mathcal{N}], K$)
    $V_t = \{(x_i, y_i) | i = 1, ..., K\}$ ▷ random sample result
    $\mathcal{D}_k = \{D_i | i \in V_t\}$ ▷ sample sub-dataset $\mathcal{D}_k$ for the task
    for each $k \in [1, K]$ **do**
      $\mathcal{S}_k$ = RandomSample($\mathcal{D}_k, N_{\mathcal{S}}$)
      $\mathcal{Q}_k$ = RandomSample($\mathcal{D}_k \backslash \mathcal{S}_k, m_q$)
      $\mathbf{C}_k = f_{\mathrm{DE}}(f_{\mathrm{EN}}(f_{\mathrm{EM}}(x_i)))$, $x_i \in \mathcal{S}_k$ ▷ representation each class
    end for
    $J = 0$
    for each $k \in [1, N_{\mathcal{S}}]$ **do**
      for each $j \in [1, m_q]$ **do**
        $J = J + L_E(f_{\mathrm{EM}}(q_j), \mathbf{C}_k) + L_R(f_{\mathrm{EM}}(q_j), \mathbf{C}_k)$
        where $q_j \in \mathcal{Q}_k$ ▷ accumulate class re-construction loss
      end for
    end for
    Back propagation
  end for
end for

B. Metric Module

Similar to other metric learning based method [6], we employ a Euclidean distance function d and produce a distribution over all classes given a sample q from the query set $\mathcal{Q}$:

$$p_E(y = k|q) = \frac{\exp(-d(f_{\mathrm{EM}}(q), \mathbf{C}_k))}{\sum_{k'} \exp(-d(f_{\mathrm{EM}}(q), \mathbf{C}_{k'}))} \tag{19}$$

As shown in Eq. 19, the distribution is based on a softmax over the distance between the embedding of the samples (in the query set) and the reconstructed features of the class. The loss function of our network can then read:

$$L_E = d(f_{\mathrm{EM}}(q), \mathbf{C}_k) + \log \sum_{k'} \exp(-d(f_{\mathrm{EM}}(q), \mathbf{C}_{k'})) \tag{20}$$

As mentioned in the main paper, a relation module is included in the metric module. The relation score RS is a $(Q_k * k) \times k$ matrix, in which the element $RS_{q,k}$ can be represented as:

$$RS_{q,k} = f_R(\varsigma(f_{\mathrm{EM}}(q), \mathbf{C}_k)), \tag{21}$$

where f_R is the relation module with two convolution layers and two fully connected layers, $\varsigma(.)$ is a function that combines

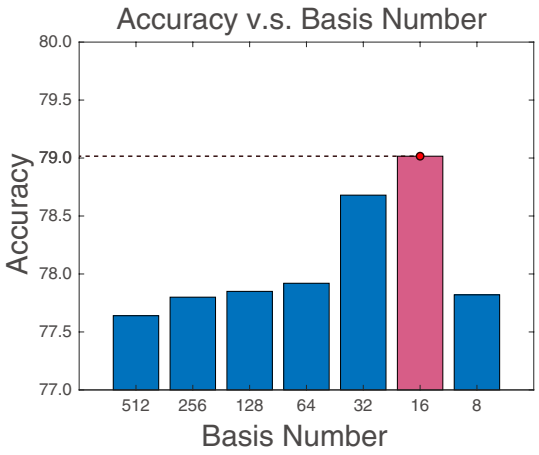

Fig. 8: Quantitative measurements of the classification accuracy against semantic basis number for class representation in class domain.

the two feature tensors. In this paper, we simply concatenate them as the input of the relation module. In [7], mean square error (MSE) is recommended to train the relation module:

$$L_{MSE} = \sum_{i=1}^{k} \sum_{q=1}^{Q_k * k} (RS_{q,k} - \mathbf{I}(k == y_q))^2, \tag{22}$$

where y_q is the correct label for query image q, $\mathbf{I}$ is a $1 * k$ dimension vector with element 1 when $k == y_q$, otherwise 0. Noted that for each query image q, the relation score RS is with the same dimension as $\mathbf{I}$, and can be treated as a probability distribution over k classes. Similar to Eq. 19, given the correct label y_q for q, the distribution can be written as:

$$p_R(y = k|q) = \frac{\exp(f_R(\varsigma(f_{\mathrm{EM}}(q), \mathbf{C}_k))}{\sum_{k'} \exp(f_R(\varsigma(f_{\mathrm{EM}}(q), \mathbf{C}_{k'})))} \tag{23}$$

Hence, the loss for the relation module can also be written as:

$$\begin{aligned} L_R &= -\log(p) \\ &= -f_R(\varsigma(f_{\mathrm{EM}}(q), \mathbf{C}_k)) + \log \sum_{k'} \exp(f_R(\varsigma(f_{\mathrm{EM}}(q), \mathbf{C}_{k'}))) \end{aligned} \tag{24}$$

The total loss of the network can then be summarized as:

$$L = \alpha_1 * L_E + \alpha_2 * L_R + \alpha_3 * \|\Theta\|^2, \tag{25}$$

where $\alpha_1, \alpha_2, \alpha_3$ are the weights of the two losses and the regularization term, Θ include the training parameters of class encoder, class decoder and metric module.

IX. Algorithmic Procedure of Class Regularization Network

The process of our proposed network can be visualized in Algorithm 1.

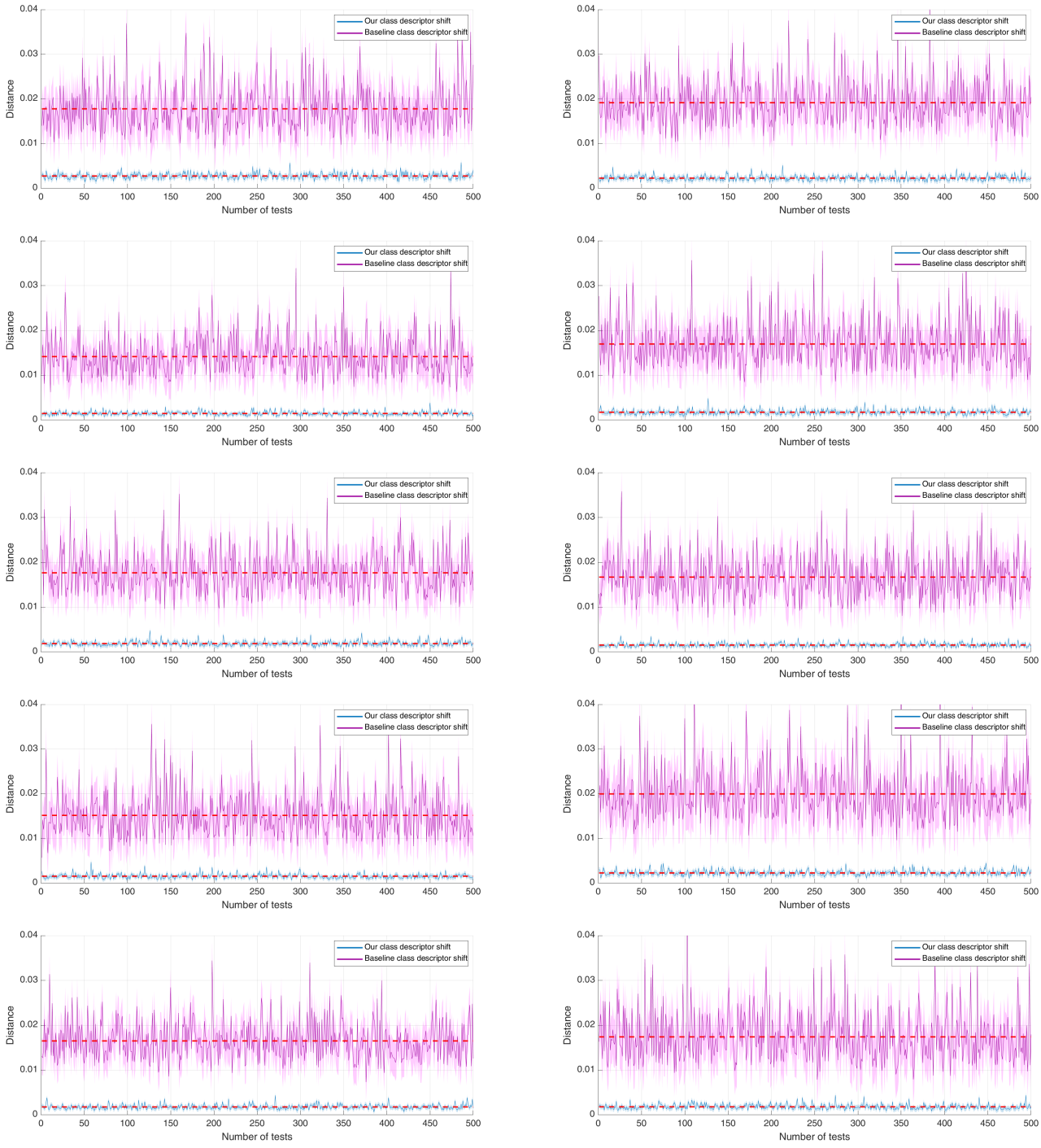

Fig. 9: Additional meta shift plots on ten different classes in test set of MiniImageNet: The statistics on the distance between class descriptors and their mean in 500 tests for baseline method (ProtoNet^{+}) and the proposed method. Class name for the descriptors: $1st$ row: 'worm', 'crab', $2nd$ row: 'golden retriever', 'malamute', $3rd$ row: 'dalmatian', 'hyena dog', $4th$ row: 'lion', 'ant', $5th$ row 'ferret', 'bookshop'.