FEW-SHOT LEARNING WITH IMPROVED LOCAL REPRESENTATIONS VIA BIAS RECTIFY MODULE

Chao Dong[1] *Qi Ye*[1] *Wenchao Meng*[1⋆] *Kaixiang Yang*[1]

[1]College of Control Science and Engineering, Zhejiang University

ABSTRACT

Recent approaches based on metric learning have achieved great progress in few-shot learning. However, most of them are limited to image-level representation manners, which fail to properly deal with the intra-class variations and spatial knowledge and thus produce undesirable performance. In this paper, we propose a Deep Bias Rectify Network (DBRN) to fully exploit the spatial information that exists in the structure of the feature representations. We first employ a *bias rectify module*, which is able to focus on the features that are more discriminative for classification by giving different weights, to alleviate the adverse impact caused by the intra-class variations. To make full use of the training data, we design a prototype augment mechanism that can make the prototypes generated from the support set to be more representative. To validate the effectiveness of our method, we conducted extensive experiments on various popular few-shot classification benchmarks and our methods can outperform state-of-the-art methods.

***Index Terms*—** Few-shot learning, attention mechanism, metric learning

1. INTRODUCTION

Few-shot learning (FSL)[1, 2, 3] aims to learn a model with good generalization capability to get rid of the dependency of the annotated data. Concretely, it can be readily adapted to the agnostic tasks with a handful of labeled examples. However, the extremely limited annotated samples per class can hardly predict the true class distribution, making FSL tasks challenging.

To tackle the FSL problem, a variety of approaches have been proposed. For example, some state-of-the-art methods resort to learning a deep embedding network to represent the inter-class diversity[4, 5, 6]. Furthermore, they use non-parametric classifiers(e.g., the nearest neighbor classifier) to avoid the complex optimization problem in learning a classifier from a few examples. Another category of methods[7, 8] constructs a meta-learner that can quickly adapt to new tasks with a few labeled samples, either by a good initialization ,or by effective learning algorithms. Moreover, the last group of methods[9, 10] directly solve the data deficiency by hallucinating new images based on labeled images from similar categories.

⋆ Corresponding author.

The previous methods mainly focus on knowledge generalization, samples generating or optimization algorithms, but have not paid sufficient attention to the way of generating appropriate representations. Such representations can make better use of the limited examples. DN4[5] goes one step further, they replace the image-level representations with the set of local representations. This approach can preserve the discriminative knowledge which loses during the global pooling process. However, this method has not taken account of the spatial information, *e.g.*, the background clutter and the intra-class variations. Such variations may force the features from the same class far away from each other in a given metric space. If there are sufficient labeled data for training, the subsequent learning procedure of convolutional neural networks with sufficient training samples can alleviate such interference. However, considering the specific nature of few-shot learning, it is nearly impossible to eliminate the impact caused by the noise and thus deteriorates the performance. Therefore, a desirable meta-learning approach should have the ability to properly utilize the spatial knowledge and reduce the interference caused by the aforementioned reasons.

A simple approach to alleviating the noise is giving higher weight to the visual features are most discriminative for a given feature map. Towards this, we propose a deep bias rectify module based on non-local attention. The module will calculate the weight of each component by comparing it with the whole representations. Besides, the bags-of-features model used in DN4[5] takes all raw features generated from the support set for classification. Such features contain a lot of noise which affects the performance. Therefore, we assembled the feature embeddings as prototypes for each visual category similar to ProtoNet[11]. In addition, we propose a simple approach called prototype to augment to reduce the scarcity of data. We resize each support image to multi-scale before sending it to the feature extractor, the output feature maps will be fused to produce a single prototype for classification.

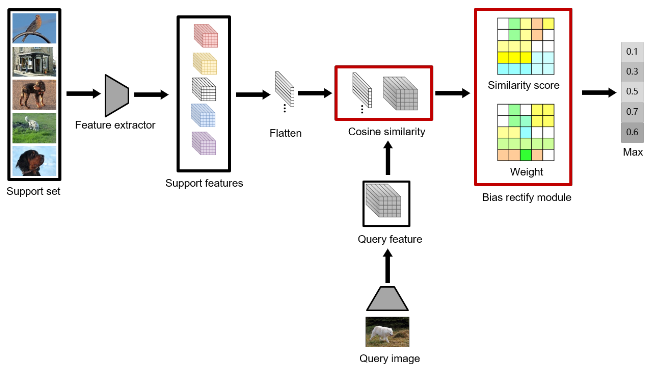

Fig. 1. Illustration of the whole network architecture of our DBRN for a 5-way 1-shot few-shot classification task.

2. OUR METHOD

2.1. Feature extractor

As the traditional set in few-shot learning, we employ a CNN as the feature extractor which represents the images from support and query set by high dimensional feature. In general, the feature extractor only contains convolutional layers but has no fully connected layers $\Psi(\cdot)$. Unlike recent literature on the few-shot classification, we remove the final global average pooling layer and produce a feature map $\Psi(x) \in \mathbb{R}^{w\times h\times d}$ rather than a vector $\Psi(x) \in \mathbb{R}^{d}$ to represent the visual feature of the image, where w denotes the width and h denotes the height of the feature map. Then the feature map of the image is formulated as:

$$\Psi(x_i) = [v_1, ..., v_r] \in \mathbb{R}^{r\times d}, (r = w \times h) \tag{1}$$

where u_i denotes the i_{th} local feature of image and r denotes the resolution of feature maps. Not in line with DN4[5] that directly uses the local features for classification, we take the centroid of the local features as the *prototype* of each class. We believe that taking the centroid approach can eliminate the bias without extra parameters, which were first proposed in ProtoNet[11]. The difference is that we take the centroid of every local feature. Based on Equation 1, the *prototype* of class c in the support set can be formulated as:

$$P_c = \frac{1}{|S_c|} \sum_{(x,y)\in S_c} \Psi(x) \in \mathbb{R}^{r\times d} \tag{2}$$

where $(x, y) \in S_c$ denotes the support examples belonging to class c, the P_c can represent the visual feature of each class in the support set. The *prototype* is used to calculate the similarity between images from the query set and support set.

2.2. Similarity module

The similarity module is used to calculate the distance between the prototype in Equation 2 and the query images to get the classification score of the query images. In this paper, we use the cosine similarity as a metric function and k-NN as a similarity function to select the similarity value for query images.

As described in Section 2.1, given a query image x, the feature extractor will embed it as $\Psi(x) = [v_1, ..., v_r]$. Then the similarity value can be calculated as

$$\begin{aligned} &Sim\left(P_c, \Psi\left(x\right)\right) = \sum_{j=0}^{k} s_\tau\left(u_i, v_j\right) \\ &s_\tau(x, y) = \tau\left\langle \hat{x}, \hat{y}\right\rangle \end{aligned} \tag{3}$$

where $P_c = [u_1, ..., u_r] \in \mathbb{R}^{r\times d}$ means the *prototype* for images belong to class c and k means we just concern the k nearest feature vectors in the *prototype* for the query image. for $x, y \in \mathbb{R}^d$, $\hat{x}$ is the l_2-normalized conuterpart for x. s_τ is the scaled cosine similarity, with τ being a learnable parameter as in [12, 13]. Then, the similarity module is defined as:

$$f_\theta[P](x) = \sigma\left(\left[Sim\left(P_c, \Psi\left(x\right)\right)\right]_{c=1}^{n}\right) \tag{4}$$

where $P = [P_1, ..., P_n]$ means the *prototype set* for all classes in support set and $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ is the *softmax function*.

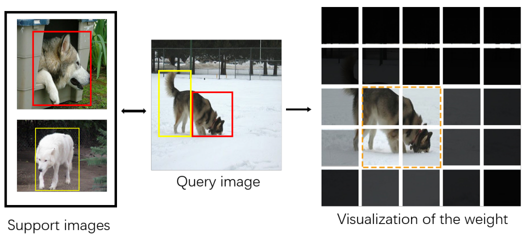

Fig. 2. The illustration for the motivation for bias rectify module.

2.3. Bias rectify module

In traditional few-shot classification tasks, the support set merely consists of several images for each class, and we need to classify different query images based on this support set. The lack of data in the support set leads to an inevitable problem: the bias in different images from the support set and the query set will deteriorate the performance of the classification model. As shown in Figure 2, different parts of the object may exist in the support images and query images, which will lead to a significant difference in the visual features. So that we propose a bias rectify module that can effectively alleviate the difference and enhance the accuracy of our model. This is the most improvement we have made based on DN4[5].

Firstly, we can observe that the features which occupy a larger area in original images will occupy more space in feature maps. Therefore, we can calculate the co-occurrence rate of the special part images by calculating their corresponding feature map co-occurrence rate. To achieve this goal, we calculate the cosine similarity between the local feature of query images and the whole feature map to generate the co-occurrence rate of this local feature:

$$W_{v_j} = \frac{1}{\xi} \sum_{i=0}^{r} \left(\frac{u_i \times v_j}{|u_i| \times |v_j|} \right)^{\omega} \tag{5}$$

where the W_{v_j} denotes the co-occurrence rate of local feature vector v_i in query images, r denotes the resolution of the feature maps generated from support images, ξ denotes the normalized parameter and ω denotes a hyper-parameter that controls the dispersion of W_{v_j}.

This bias rectify module is simple but effective during few-shot classification tasks that can benefit the experiment result without any extra parameters. This module will be more effective when the support images and query images have a different appearance.

2.4. Prototype augment

Considering the lack of labelled data, the size of the object in the support images may differ from the object in the query images. Therefore, we resize the support images in multi-scale before inputting them to the feature extractor, then we calculate the prototypes of the support set with all these feature maps. By this method, the prototypes of the support set become more robust when encountering the variation of the size of the object. The prototype generated from the support set can be represented as

$$Prototype(c) = \frac{1}{|S_c|} \sum_{P_{c_i} \in S_c} P_{c_i} \tag{6}$$

where $S_c = [P_{c_i}, ..., P_{c_n}]$ denotes the set of prototypes with different scale, P_{c_i} denote the *prototype* of class c with the $i-th$ scale which can be calculated by Equation 2. In our experiments, we take the triple-scale method, which means the images in the support and query set will become three times during meta-training and testing. During experiment, we set the resolution of the images as 84×84, 92×92, 108×108.

3. EXPERIMENTS

3.1. Dataset

We use the ResNet-12 trained following previous work FEAT[14] on a RTX3090. We evaluate the performance of our method for few-shot classification tasks on three popular benchmark dataset: miniImageNet[4], tieredImageNet[15], Caltech-USCD birds-200-2011(CUB)[16].

*mini*ImageNet is derived from ILSVRC-12 dataset. It contains 100 different classes with 600 samples per class. We follow the splits used in previous work [17], which divide the dataset into 64, 16, 20 for training, validation and test, respectively. *tiered*ImageNet is a larger subset derived from ILSVRC-12 dataset[18]. It contains 608 classes from 34 super-classes, which includes 1281 images for each class. We follow the splits in [15], which take 351, 97 and 160 classes as the training set, validation set and testing set respectively. CUB was originally proposed for fine-grained classification tasks. It includes 200 different birds with 11,788 images in total. Following the splits in previous works [19], we take 100 classes for training, 50 classes for validation and 50 classes for testing.

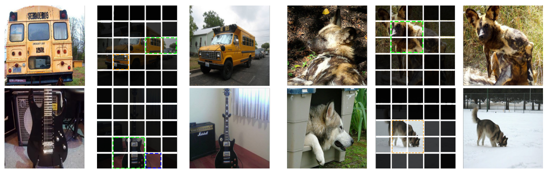
Fig. 3. Visualization for our DBRN model.

3.2. Implementation details

Because the motivation of this work is largely inspired by the DN4 [5]. We re-implement the DN4 as our baseline model. We use the ResNet12[13] as our model backbone following the previous literature to get a fair comparison with previous work. For each dataset, we pre-train a classifier with the training set. Then we remove the fully connected layer in ResNet12, so that the network becomes a convolutional network that maps each input image as a feature vector. In order to preserve the local feature for input images, we remove the global average pooling layer so that the network becomes a fully convolutional network. When input images resize as 84×84, the backbone network generates a feature map with size $5 \times 5 \times 640$. Different from previous works which train the network from scratch, we apply a pre-train strategy as suggested in [20]. After pre-training, we remove the fully connected layer and select the pretrained model with the highest performance in validation set.

3.3. Main results and Comparisons

Table 1 and Table 2 present 5-way classification accuracy (%) with 95% confidence intervals of our method and others on miniImageNet, tieredImageNet and CUB. We take the *k*-NN algorithm similar to DN4[5] as our baseline, which selects the k nearest feature vectors between support set and query image as similarity vector. Firstly, we can observe that our baseline already archives a better performance than some state-of-the-art methods. This may be because we take an end-to-end training strategy rather than fixing the backbone network and temperature scaling of the logits while meta-training. Moreover, our DBRN further promotes the performance and outperforms all state-of-the-art methods with a significant margin which effectively demonstrate the effectiveness of our method. By comparing the performance of our DBRN and our baseline,

Table 1. Average 5-way classification performance(%) with 95% confidence intervals on *mini*ImageNet and *tiered*ImageNet.

Method	backbone	***mini*ImageNet**		***tiered*ImageNet**	
		1-shot	5-shot	1-shot	5-shot
ProtoNet[11]	*ResNet12*	60.37 ± 0.83	78.02 ± 0.57	65.65 ± 0.92	83.40 ± 0.65
Baseline++[19]	*ResNet12*	55.43 ± 0.81	77.18 ± 0.61	61.49 ± 0.91	82.37 ± 0.67
Neg-Cosine[21]	*ResNet12*	63.85 ± 0.81	81.57 ± 0.56	-	-
FEAT[14]	*ResNet12*	66.78 ± 0.20	82.05 ± 0.15	70.80 ± 0.23	84.79 ± 0.16
CTM[22]	*ResNet12*	64.12 ± 0.82	80.51 ± 0.13	68.41 ± 0.39	84.28 ± 1.73
DeepEMD[23]	*ResNet12*	65.91 ± 0.82	82.41 ± 0.56	71.16 ± 0.87	86.03 ± 0.58
E3BM[24]	*ResNet12*	63.80 ± 0.40	80.10 ± 0.30	71.20 ± 0.40	85.30 ± 0.30
PPA[20]	*WRN-28-10*	$59.60 \pm -$	$77.46 \pm -$	$63.99 \pm -$	$81.97 \pm -$
LEO[25]	*WRN-28-10*	61.76 ± 0.08	77.59 ± 0.12	66.33 ± 0.05	81.44 ± 0.09
TADAM[26]	*ResNet12*	58.50 ± 0.30	76.70 ± 0.30	-	-
Our Baseline	*ResNet12*	61.13 ± 0.27	76.97 ± 0.20	70.45 ± 0.22	83.37 ± 0.33
DBRN(Ours)	*ResNet12*	**67.01 ± 0.28**	**83.33 ± 0.19**	**72.80 ± 0.31**	**87.13 ± 0.21**

Table 2. Average 5-way classification performance(%) with 95% confidence intervals on CUB.

Method	backbone	**CUB**	
		1-shot	5-shot
ProtoNet[11]	*ResNet12*	66.09 ± 0.92	82.50 ± 0.58
Baseline++[19]	*ResNet12*	67.02 ± 0.90	83.58 ± 0.54
Neg-Cosine[21]	*ResNet12*	72.66 ± 0.85	89.40 ± 0.43
MAML[7]	*Conv4*	50.45 ± 0.97	59.60 ± 0.84
MVT[27]	*ResNet12*	-	80.33 ± 0.61
DeepEMD[23]	*ResNet12*	75.65 ± 0.83	88.69 ± 0.50
Our Baseline	*ResNet12*	66.63 ± 0.28	81.64 ± 0.19
DBRN(Ours)	*ResNet12*	**75.78 ± 0.27**	**92.21 ± 0.14**

Table 3. Ablation study on miniImageNet with 95% confidence intervals.

Pow	Weight	ProtoAug	*mini*ImageNet	
			5way1shot	5way5shot
✗	✗	✗	63.88	80.52
✗	✓	✗	65.53	81.80
✓	✓	✗	66.03	82.58
✓	✓	✓	67.01	83.33

we can observe that applying an original k-NN strategy on the backbone for the few-shot classification is not enough. It means that simply representing similarity for two images with the nearest features discards important knowledge between images.

3.4. Ablative study

To further validate the effectiveness of our method, we apply ablation experiments as shown in Table 3. Weight and pow are components for the bias rectify module in Equation 5 . Protoaug is prototype augment mechanism in Equation 6. The ablation of each component in our model will result in the drop of performance.

3.5. Visualization of bias rectify weights

To demonstrate the effectiveness of the bias rectify module proposed in Section 2.3 during the inference process, we conduct a visualization experiment between regions of correspondence images for support set and query set. We give different brightness for the regions of query images concerning their weight. As it is shown in Figure 3 (Please zoom it for details), the regions which co-occurrence in both images will get higher weight, regardless of the size difference among these regions. That means bias rectify module will alleviate the bias among different images and benefit the classification performance. We take both correctly and incorrectly classified pair of images in our experiment to further demonstrate the robustness of our method.

4. CONCLUSION AND FUTURE WORK

In this work, we proposed a simple but effective DBRN for the few-shot classification. We used feature extractors based on end-to-end training on base-class with standard cross-entropy loss without any extra data or complex meta-training strategies that were advocated in recent few-shot literature. Firstly, we focus on the intra-class variations and high-variance background that exists in the support and query set and proposed a nonparametric rectify module to alleviate the influence of this bias. Then prototype augment mechanism is proposed for the lack of the labeled data during inference. In this context, designing a more powerful prototype generation mechanism that can effectively utilize the discriminative information between images looks like a very promising direction for future research. The experiment results obtained on three popular datasets demonstrated that our DBRN significantly outperforms existing methods and achieves new state-of-the-art on few-shot classification task.

5. REFERENCES

[1] Aming Wu, Yahong Han, Linchao Zhu, Yi Yang, and Cheng Deng, "Universal-prototype augmentation for few-shot object detection," *CoRR*, vol. abs/2103.01077, 2021.

[2] Suiyi Ling, Andreas Pastor, Jing Li, Zhaohui Che, Junle Wang, Jieun Kim, and Patrick Le Callet, "Few-shot pill recognition," in *CVPR*. IEEE, 2020, pp. 9786–9795.

[3] Yaqing Wang, ABULIKEMU ABUDUWEILI, quanming yao, and Dejing Dou, "Property-aware relation networks for few-shot molecular property prediction," in *Advances in Neural Information Processing Systems*, 2021.

[4] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra, "Matching networks for one shot learning," in *NIPS*, 2016, pp. 3630–3638.

[5] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Gao Yang, and Jiebo Luo, "Revisiting local descriptor based image-to-class measure for few-shot learning," in *CVPR*. IEEE, 2019, pp. 7253–7260.

[6] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *CVPR*. IEEE, 2018, pp. 1199–1208.

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

[8] Antreas Antoniou, Harrison Edwards, and Amos Storkey, "How to train your MAML," in *ICLR*, 2019.

[9] Bharath Hariharan and Ross B. Girshick, "Low-shot visual recognition by shrinking and hallucinating features," in *CVPR*. 2017, pp. 3037–3046, IEEE Computer Society.

[10] Yu-Xiong Wang, Ross B. Girshick, Martial Hebert, and Bharath Hariharan, "Low-shot learning from imaginary data," in *CVPR*. 2018, pp. 7278–7286, IEEE Computer Society.

[11] Jake Snell, Kevin Swersky, and Richard S. Zemel, "Prototypical networks for few-shot learning," in *NIPS*, 2017, pp. 4080–4090.

[12] Hang Qi, Matthew Brown, and David G. Lowe, "Low-shot learning with imprinted weights," in *CVPR*. 2018, pp. 5822–5830, IEEE Computer Society.

[13] Spyros Gidaris and Nikos Komodakis, "Dynamic few-shot visual learning without forgetting," in *CVPR*. 2018, pp. 4367–4375, IEEE Computer Society.

[14] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha, "Few-shot learning via embedding adaptation with set-to-set functions," in *CVPR*. 2020, pp. 8805–8814, IEEE.

[15] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel, "Meta-learning for semi-supervised few-shot classification," in *ICLR*, 2018.

[16] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," *california institute of technology*, 2011.

[17] Sachin Ravi and Hugo Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2017.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[19] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang, "A closer look at few-shot classification," in *ICLR*, 2019.

[20] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L. Yuille, "Few-shot image recognition by predicting parameters from activations," in *CVPR*. 2018, pp. 7229–7238, IEEE Computer Society.

[21] 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 *ECCV*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, Eds. 2020, vol. 12349, pp. 438–455, Springer.

[22] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang, "Finding task-relevant features for few-shot learning by category traversal," in *CVPR*. 2019, pp. 1–10, Computer Vision Foundation / IEEE.

[23] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen, "Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers," in *CVPR*. 2020, pp. 12200–12210, IEEE.

[24] Yaoyao Liu, Bernt Schiele, and Qianru Sun, "An ensemble of epoch-wise empirical bayes for few-shot learning," in *ECCV*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, Eds. 2020, vol. 12361, pp. 404–421, Springer.

[25] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell, "Meta-learning with latent embedding optimization," in *ICLR*, 2019.

[26] Boris N. Oreshkin, Pau Rodríguez López, and Alexandre Lacoste, "TADAM: task dependent adaptive metric for improved few-shot learning," in *NeurIPS*, 2018, pp. 719–729.

[27] Seong-Jin Park, Seungju Han, Ji-Won Baek, Insoo Kim, Juhwan Song, Haebeom Lee, Jae-Joon Han, and Sung Ju Hwang, "Meta variance transfer: Learning to augment from the others," in *ICML*. 2020, vol. 119, pp. 7510–7520, PMLR.