Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Self-reconstruction network for fine-grained few-shot classification

Xiaoxu Li [a,b], Zhen Li [a], Jiyang Xie [b], Xiaochen Yang [c,*], Jing-Hao Xue [d], Zhanyu Ma [b]

[a] School of Computer and Communication, Lanzhou University of Technology, Lanzhou, 730050, China
[b] Pattern Recognition and Intelligent System Laboratory, School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing, 100876, China
[c] School of Mathematics and Statistics, University of Glasgow, Glasgow, G12 8QQ, UK
[d] Department of Statistical Science, University College London, London, WC1E 6BT, UK

ARTICLE INFO

Keywords:
Few-shot learning
Fine-grained image classification
Deep neural network
Self-reconstruction network

ABSTRACT

Metric-based methods are one of the most common methods to solve the problem of few-shot image classification. However, traditional metric-based few-shot methods suffer from overfitting and local feature misalignment. The recently proposed feature reconstruction-based approach, which reconstructs query image features from the support set features of a given class and compares the distance between the original query features and the reconstructed query features as the classification criterion, effectively solves the feature misalignment problem. However, the issue of overfitting still has not been considered. To this end, we propose a self-reconstruction metric module for diversifying query features and a restrained cross-entropy loss for avoiding over-confident predictions. By introducing them, the proposed self-reconstruction network can effectively alleviate overfitting. Extensive experiments on five benchmark fine-grained datasets demonstrate that our proposed method achieves state-of-the-art performance on both 5-way 1-shot and 5-way 5-shot classification tasks. Code is available at https://github.com/liz-lut/SRM-main.

1. Introduction

Deep learning has achieved impressive performance in computer vision. However, deep networks usually demand numerous labeled data for training, which is impractical in many tasks where data acquisition is costly and time-consuming. For this reason, researchers in the computer vision community turn considerable attention to few-shot learning in recent years, especially for few-shot classification [1–3].

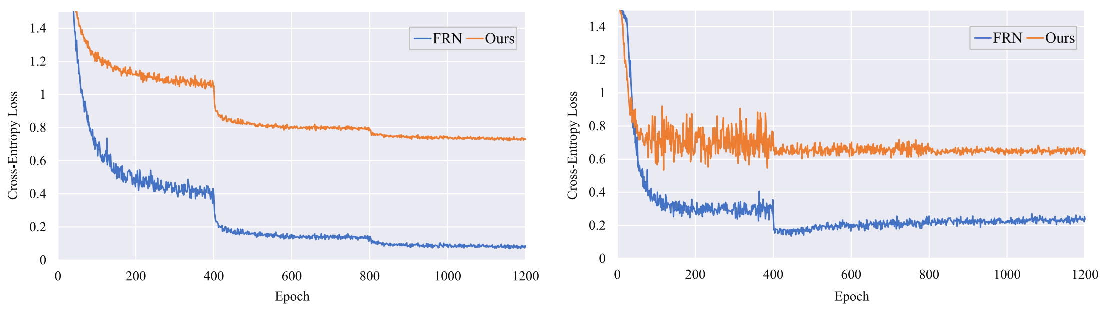
The goal of few-shot classification is to recognize unseen query sample with very limited (often less than 10) labeled support samples. Existing methods usually can be categorized into three classes, metric-based methods [4], optimization-based methods [5], and transfer learning-based methods [6,7]. Among these methods, metric-based methods are relatively simple but effective, achieving state-of-the-art performance in many few-shot tasks. Metric-based methods usually adopt episodic training strategy to train a feature extractor with a fixed distance metric or a parameterized distance metric, and then fix them, i.e., without fine-tuning, to classify unseen novel query samples. However, early works, e.g., prototypical network [8] and relation network [9], mainly build on global features, which suffer from inaccurate similarity measure between two samples due to the mismatch of key information in images. This is particularly detrimental to few-shot fine-grained image classification, as sub-categories have subtle differences and the valuable and discriminative information is likely to locate in different regions. To address such an issue, some subsequent works start to focus on learning a metric on local features [10] or aligning local features [11,12].

Recently, some metric-based approaches introducing new alignment [12] or reconstruction [13] techniques have achieved impressive performance in fine-grained few-shot image classification. However, in the experiment, we noticed that the state-of-the-art method, feature reconstruction network (FRN) [13], suffered from overfitting during episodic training. As shown in Fig. 1, while the loss function of FRN keeps decreasing on the training set, it increases on the validation set after 400 epochs. This overfitting phenomenon may occur as, comparing with that of ImageNet, CIFAR, etc., the numbers of images and classes in fine-grained datasets are relatively small and thus the task diversity in episodic training is limited.

To alleviate overfitting, we propose a self-reconstruction network for few-shot fine-grained classification, which introduces a new self-reconstruction metric module and a restrained cross-entropy loss. The self-reconstruction metric module not only reconstructs query features from support features as in FRN, but more importantly, it also reconstructs query features from themselves. Such self-reconstruction can effectively augment and diversify query features without introducing artifacts, and as shown in the experiment, it avoids over-reliance on

* Corresponding author.
E-mail address: xiaochen.yang@glasgow.ac.uk (X. Yang).

https://doi.org/10.1016/j.patcog.2024.110485
Received 16 March 2023; Received in revised form 10 October 2023; Accepted 5 April 2024
Available online 6 April 2024



(a) Training loss

(b) Validation loss

Fig. 1. Motivation of the self-reconstruction network. FRN [13] encounters the overfitting problem as its training loss keeps decreasing but its validation loss starts to increase after around 400 epochs. In contrast, the validation loss of our method stays stable after 400 epochs, demonstrating its effectiveness in addressing overfitting.

one discriminative feature. After feature reconstruction, both the distance between the original query features and its support-reconstructed features and the distance between the support-reconstructed and self-reconstructed query features are calculated, whose sum is used to match a query sample and the support class. The self-reconstruction network is trained according to the classical cross-entropy loss and a new restrained cross-entropy loss. The latter can prevent the model from producing over-confident predictions which were due to overfitting the training data. By utilizing self-reconstruction and the restrained cross-entropy loss, the proposed method can avoid overfitting, as shown in Fig. 1.

To summarize, the contributions of our work are three-fold:

1. We are the first to propose using self-reconstruction to increase the diversity of query features, which can effectively expand the representation capability of the learned query feature space and mitigate the overfitting problem.
2. We further propose a restrained cross-entropy loss, which can be easily equipped with existing metric-based few-shot classification models.
3. Experiments on five fine-grained datasets demonstrate the superiority of the proposed method, with detailed ablation studies showing that both self-reconstruction and restrained loss are effective in alleviating overfitting.

2. Related work

In this section, we first provide a concise review on fine-grained few-shot learning methods. Next, we review two types of methods that are most relevant to this work, namely metric-based methods and alignment-based methods. For a more comprehensive review on fine-grained few-shot classification, we refer readers to [14].

2.1. Fine-grained few-shot classification

Fine-grained few-shot classification faces the dual challenges of scarce labeled data and the subtle distinctions between different subcategories, e.g., distinguishing between beagle and pug within the category of dog. Global features, which capture image-level concepts, are insufficient to discriminate between fine-grained categories. Therefore, a line of research focus on local features. DN4 [10], a notable method for fine-grained classification, first utilized intermediate levels of CNN as local features and performed classification according to the aggregated distances calculated over local features and its k-nearest neighbors. As an improvement of DN4, LSANet [15] allowed for different scales of local patches to better capture the structure information and assigned different weights to query patches to suppress the background and highlight the targets. MCL-Katz [16] aggregated local features into global features using weights set as the stationary distribution of local features. AGPF-FSFG [17] constructed multi-scale features and reweighted them via multi-level attention. Our method also makes use of local features to build the support set or query set, which are then used to generate support-reconstructed query features and self-reconstructed query features.

In addition, our method shares the idea of data augmentation. Methods such as Hallucinator [18] and FOT [19] assume that variations in illumination, backgrounds or poses can be shared across classes and thus can be utilized to diversify the limited support samples. In contrast, our work focuses on diversifying the query samples, and we employ the ridge regression technique for efficient self-reconstruction, eliminating the necessity to model intra-class variations.

2.2. Metric-based methods

Metric-based methods constitute one mainstream approach in few shot learning. These methods learn a transferable feature embedding network such that queries can be classified according to the similarity between query features and support features. The similarity can be predefined such as by using cosine similarity [20] or learned via a neural network [9]. A pioneering metric-based method is the prototypical network [8], which first constructs prototypes as the average of support features and then compares queries with these class representations. To adapt to fine-grained classification, LMPNet [21] used multiple prototypes per class and constructed prototypes as weighted averages of feature embeddings with learnable weights. COMET [22] learned multiple embedding functions, one for each image segment or concept, and accordingly constructed multiple concept prototypes. PHR [23] learned feature embeddings at local, global, and semantic levels and updated prototypes according to novel data. SAPENet [24] obtained more representative prototypes by emphasizing discriminative local features and channels using self-attention and the proposed intra-class attention, respectively.

Another direction of development in metric-based methods is on the distance measure itself. To list a few, Deep BDC [25] proposed the Brownian distance covariance metric to exploit the joint distributions between support and query features. BSNet [26] combined cosine similarity and relation score for learning more discriminative features in fine-grained images. Temperature Network [27], despite using a single similarity measure, gradually tuned the temperature scaling parameter in the measure, which acts similarly to enforcing a large-margin metric. Different from these methods, our method combines two Euclidean distances — one is between the original query and the support-reconstructed query, and the other is between the self-reconstructed query and the same set of support-reconstructed query.

2.3. Alignment-based methods

One issue with metric-based methods is that position information of the embedded features of labeled samples may not correspond to that of unseen samples and therefore the distance calculated directly over these features can be very large, even for samples from the same category. To this end, alignment-based methods have been proposed [12, 13,28]. LRPABN [28] trained a position transformation matrix to rearrange the position of support local features to match the query ones. DeepEMD [12] addressed the spatial inconsistency by adopting the earth mover's distance, which can be interpreted as the optimal matching cost of aligning two sets of local features extracted from a support and a query image. FRN [13] reconstructed query features from support features based on ridge regression, which avoids introducing many parameters as in aforementioned methods and admits a closed-form solution. Building on FRN, LCCRN [29] improved the features by utilizing information from neighborhood pixels. The resulting features were used to construct four cross-reconstruction tasks, whose reconstruction errors were combined using learnable weights. Besides spatial alignment, channel alignment has also been considered [30–32]. TDM [30] performed channel alignment, which used attention on the support set to highlight class-wise discriminative channels and on a query instance to highlight object-relevant channels. SaberNet [31] adopted Swin Transformer as the feature extractor to capture spatial long-range dependencies between local features and aligned query features and refined prototype features at both spatial and channel levels.

In this work, we perform feature reconstruction in a similar manner to FRN [13] by adopting the ridge regression. However, our method self-reconstructs the query features from themselves, which diversifies the query features without introducing artifacts. Consequently, the representation capability of the learned query feature space is expanded, thus alleviating overfitting and leading to better generalization.

3. Self-reconstruction network

3.1. Problem definition

The few-shot classification problem usually divides a given dataset into three sub-datasets according to different stages. D_{train} is used for model training, D_{val} is used for model evaluation in the training stage, and D_{test} is used for the final test of the trained model. The three sub-datasets contain different image categories.

N-way K-shot is a common setting for few-shot classification, which means that a classification task consists of N classes and each class has K labeled samples. At different stages, each sub-dataset (D_{train}, D_{val} and D_{test}) is used to construct a series of tasks and each task contains a support set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{n}$ $(n = N \times K)$ and a query set $\mathcal{Q} = \{(x_q, y_q)\}_{q=1}^{m}$ $(m = N \times M)$, where x denotes the image and y denotes the class label. The support set is formed by first randomly selecting N classes and then randomly selecting K images for each of these N classes. The query set is formed by randomly selecting M images for the same N classes. Features and/or metrics are learned from the labeled support set and used to perform classification on the unlabeled query set.

3.2. Feature reconstruction by ridge regression

Ridge regression is one of the most widely-used penalized regression methods for analyzing multivariate data with multicollinearity. FRN [13] adopted this technique to reconstruct the features to solve the few-shot image classification task and achieved state-of-the-art results. For this reason, we also propose our model based on the strategy of ridge regression.

Ridge regression shrinks the coefficient estimates by adding a penalty on squared coefficient values, i.e., by minimizing the following penalized residual sum of squares:

$$\hat{\boldsymbol{\beta}}^{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} \{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|^2 \}, \tag{1}$$

where $\mathbf{y}$ is the response vector, $\mathbf{X}$ is the design matrix, $\boldsymbol{\beta}$ is the coefficient vector, and λ is a parameter controlling the magnitude of the penalty.

In the case of feature-map reconstruction here, the response is a matrix. Therefore, the coefficient should also be a matrix and, following the idea of ridge regression, the objective function is revised as follows:

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \{ \|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2 + \lambda \|\mathbf{A}\|_F^2 \}, \tag{2}$$

where $\mathbf{Y}$ denotes the response matrix, $\mathbf{A}$ denotes the coefficient matrix, and $\|\cdot\|_F$ denotes the Frobenius norm. The optimal solution is given by

$$\hat{\mathbf{A}}(\lambda) = \mathbf{Y}\mathbf{X}^\top \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I} \right)^{-1}. \tag{3}$$

The most expensive cost in calculating Eq. (3) is the inverse operation, which is $\mathcal{O}(q^3)$ for an $q \times p$ matrix $\mathbf{X}$. When $p < q$, it is computationally more efficient to calculate the following equivalent solution, which is obtained by applying the Woodbury matrix identity [33]:

$$\hat{\mathbf{A}}(\lambda) = \mathbf{Y} \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I} \right)^{-1} \mathbf{X}^\top. \tag{4}$$

The computational cost of Eq. (4) is $\mathcal{O}(p^3)$, which is smaller than that of Eq. (3) when $p < q$.

3.3. Architecture overview

Our network structure, as shown in Fig. 2, mainly consists of three modules. The first module is the feature extraction module, which maps the original image to the embedding feature. The second is the self-reconstruction metric module (SRM), which feeds the support features and a query feature into the feature reconstruction modules (RMs) to obtain two types of reconstructed query features, one reconstructed by the support features and the other self-reconstructed by the query feature. Then the distance between the support-reconstructed query features and the original feature and the distance between the support-reconstructed query features and the self-reconstructed query feature are measured separately. The third is a joint loss module based on the output distances. In the training phase, the network is optimized according to the joint loss, which includes the proposed restrained cross-entropy loss L_{RCE} and the standard cross-entropy loss L_{CE}. The pseudo algorithm of our method is provided in Algorithm 1.

3.4. Self-reconstruction metric module (SRM)

In this paper, we propose a self-reconstruction metric module, which reconstructs query features from both support features and from the query features themselves while enabling both sides to be metricized. Reconstructing query features from support features could achieve spatial alignment between query and support, and reconstructing query features from themselves could augment query features and increase task diversity.

In the N-way K-shot setting, the feature extraction module outputs feature maps $\mathcal{S}_c$ and $\mathbf{Q}_q$, where $\mathcal{S}_c$ contains features from K support samples of class $c \in C$ and $\mathbf{Q}_q$ $(q = 1, \ldots, m)$ is the feature of a single query sample. Moreover, we pool all features from the same class, i.e., applying a reshaping function to $\mathcal{S}_c$ to map all features of class c into a single matrix $\mathbf{S}_c$: $\mathbb{R}^{K \times hw \times l} \to \mathbb{R}^{Khw \times l}$, where h, w, l are the height, width and number of channels of the feature map respectively.

One core component of the SRM module is the reconstruction of query features, using both support features and their own query features, as elaborated below.

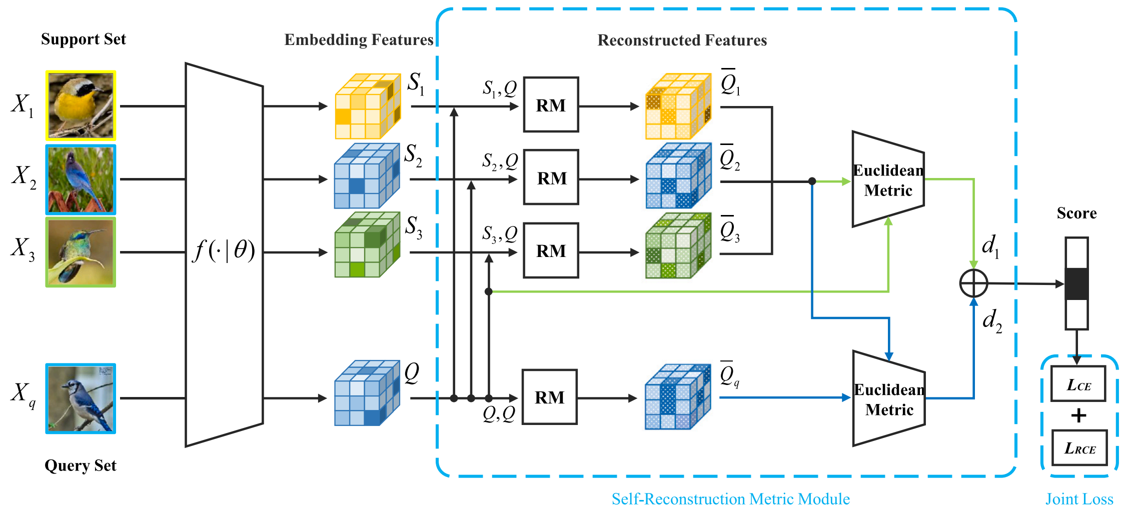


Fig. 2. The model architecture of the self-reconstruction network. Support and query images are mapped to the feature space using $f(\cdot|\theta)$. Next, these features are sent to the proposed self-reconstruction metric module, which generates reconstructed features through the reconstruction module RM and calculates the distances. The network is trained according to the proposed joint loss, which combines the cross entropy loss (L_{CE}) and restricted cross entropy loss (L_{RCE}).

Algorithm 1 Training procedure of self-reconstruction network for N-way K-shot classification

```
Input: training data $D_{train}$, number of classes $N$, number of support images
    per class $K$, number of query images per class $M$, number of episodes $t$,
    optimizer (SGD).
Output: model parameter $\theta$.
1:  for $e$=1 to $t$ do
2:      Sample a task $\mathcal{T}$ from $D_{train}$;
3:      Split $\mathcal{T}$ into the support set $\mathcal{S}$ and the query set $\mathcal{Q}$;
4:      Use $f(\cdot|\theta)$ to extract class-specific features $\{\mathbf{S}_c\}_{c=1}^{N}$ and query features $\{\mathbf{Q}_q\}_{q=1}^{NM}$;
5:      for $\mathbf{Q}_q$ in $\{\mathbf{Q}_q\}_{q=1}^{NM}$ do
6:          Compute the class-specific reconstructed query feature $\bar{\mathbf{Q}}_c$ using Eq. (5);
7:          Compute the self-reconstructed query feature $\bar{\mathbf{Q}}_q$ using Eq. (6);
8:          Compute the squared Euclidean distance $d_{c,1}$ between the $\bar{\mathbf{Q}}_c$ and $\mathbf{Q}_q$ using Eq. (9);
9:          Compute the squared Euclidean distance $d_{c,2}$ between the $\bar{\mathbf{Q}}_c$ and $\bar{\mathbf{Q}}_q$ using Eq. (10);
10:         Obtain the distance between the query and the class $c$: $d_{c,q} = d_{c,1} + d_{c,2}$;
11:         Obtain the final classification probability $P$ using Eq. (12);
12:         Compute the loss of model $Loss$ using Eq. (15);
13:         Update model parameter $\theta$ to minimize $Loss$ using the optimizer.
```

The feature map $\mathbf{Q}_q$ can be reconstructed from support features $\mathbf{S}_c$ according to Eq. (4), generating the class-specific reconstructed query features $\bar{\mathbf{Q}}_c$:

$$\bar{\mathbf{Q}}_c = \rho\mathbf{A}(\lambda)\mathbf{S}_c = \rho\mathbf{Q}_q\left(\mathbf{S}_c^\top\mathbf{S}_c + \lambda\mathbf{I}\right)^{-1}\mathbf{S}_c^\top\mathbf{S}_c, \tag{5}$$

where ρ is a learnable re-scaling parameter. Eq. (5) addresses the misalignment between query and support features.

Applying the same technique, we can reconstruct $\mathbf{Q}_q$ from its own feature, so as to map the query feature to a reconstruction space. Note that the feature map $\mathbf{Q}_q \in \mathbb{R}^{hw\times l}$ comes from a single query image, rather than all query images. The self-reconstructed query feature $\bar{\mathbf{Q}}_q$ is obtained as follows:

$$\bar{\mathbf{Q}}_q = \rho\mathbf{A}(\lambda)\mathbf{Q}_q = \rho\mathbf{Q}_q\left(\mathbf{Q}_q^\top\mathbf{Q}_q + \lambda\mathbf{I}\right)^{-1}\mathbf{Q}_q^\top\mathbf{Q}_q. \tag{6}$$

The parameters ρ and λ in Eqs. (5) and (6) are not shared and they are updated in the same way as parameters in the feature extraction module during training. In order to ensure ρ and λ are positive, they are converted to e^α and e^β respectively, with α and β initialized to zero:

$$\lambda = \frac{Khw}{l}e^\alpha, \tag{7}$$

$$\rho = e^\beta. \tag{8}$$

After obtaining the reconstructed features, we carry out distance calculation as in metric-based methods. In this step, the class-specific reconstructed query features $\bar{\mathbf{Q}}_c$ and the self-reconstructed query features $\bar{\mathbf{Q}}_q$ are used differently, where $\bar{\mathbf{Q}}_c$ serves as class-specific prototypes for spatially aligning support features with the original query features $\mathbf{Q}_q$ and $\bar{\mathbf{Q}}_q$ serves as additional samples for expanding the representation capability of the learned query feature space. More specifically, we calculate the squared Euclidean distance $d_{c,1}$ between the support-reconstructed query features $\bar{\mathbf{Q}}_c$ and the original query feature $\mathbf{Q}_q$, the squared Euclidean distance $d_{c,2}$ between $\bar{\mathbf{Q}}_c$ and the self-reconstructed query feature $\bar{\mathbf{Q}}_q$, and finally sum up the two distances to get the distance $d_{c,q}$, which represents the distance between the query and the class c:

$$d_{c,1} = \|\bar{\mathbf{Q}}_c - \mathbf{Q}_q\|_F^2, \tag{9}$$

$$d_{c,2} = \|\bar{\mathbf{Q}}_c - \bar{\mathbf{Q}}_q\|_F^2, \tag{10}$$

$$d_{c,q} = d_{c,1} + d_{c,2}. \tag{11}$$

3.5. Loss functions

By using the above distance, the final classification probability can be obtained as:

$$P\left(y_q = c \mid x_q\right) = \frac{\exp\left(-\tau d_{c,q}\right)}{\sum_{c'\in C}\exp\left(-\tau d_{c',q}\right)}, \tag{12}$$

where τ is a learnable hyperparameter that controls the sharpness of the metric distance.

In the training phase, we use a joint loss integrating two loss functions to optimize the model. One is the widely-used cross-entropy (CE) loss:

$$L_{CE} = -\frac{1}{m}\sum_{q=1}^{m}\left(\boldsymbol{y}_q^\top \log\left(\boldsymbol{p}_q\right)\right), \tag{13}$$

where $\boldsymbol{y}_q$ denotes the one-hot vector, $\boldsymbol{p}_q$ denotes the vector of predicted probability, and m is the number of query samples.

To further alleviate the overfitting problem, we propose to restrain the cross-entropy loss on the training classes. Specifically, we design a new restrained cross-entropy (RCE) loss in Eq. (14), which has the

Table 1
Comparison of few-shot classification methods on CUB-200–2011, Flowers, FGVC Aircraft, Stanford-Cars, and Stanford-Dogs datasets. All experiments adopt ResNet-12 as the backbone network. Mean accuracy and 95% confidence interval are reported. The best-performing methods are shown in bold and the second best ones are underlined.

Method	CUB	Flowers	Aircraft	Cars	Dogs
	5-Way 1-Shot Accuracy (%)				
MatchingNet (NeurIPS 16')[20]	73.02 ± 0.88	75.70 ± 0.88	82.2 ± 0.80	75.70 ± 0.88	66.48 ± 0.88
ProtoNet (NeurIPS 17')[8]	79.64 ± 0.20	75.41 ± 0.22	86.57 ± 0.18	82.29 ± 0.20	72.85 ± 0.22
RelationNet (CVPR 18')[9]	63.94 ± 0.92	69.51 ± 1.01	74.2 ± 1.04	46.04 ± 0.91	47.35 ± 0.88
Baseline++ (CVPR 19')[1]	64.62 ± 0.98	69.03 ± 0.92	74.51 ± 0.90	67.92 ± 0.92	59.64 ± 0.89
DeepEMD (CVPR 20')[12]	71.11 ± 0.31	70.00 ± 0.35	69.86 ± 0.30	73.30 ± 0.29	67.59 ± 0.30
VFD (ICCV 21')[35]	79.12 ± 0.83	76.20 ± 0.92	76.88 ± 0.85	77.52 ± 0.85	76.24 ± 0.87
FRN (CVPR 21')[13]	83.16 ± 0.19	81.07 ± 0.20	88.04 ± 0.17	86.48 ± 0.18	76.49 ± 0.21
TDM (CVPR 22')[30]	82.41 ± 0.19	82.85 ± 0.19	88.35 ± 0.17	87.04 ± 0.17	76.20 ± 0.21
MCL-Katz (CVPR 22')[16]	**85.63 ± 0.00**	76.55 ± 0.00	87.69 ± 0.00	85.04 ± 0.00	71.49 ± 0.00
DeepBDC (CVPR 22')[25]	79.32 ± 0.43	80.57 ± 0.49	83.14 ± 0.41	83.24 ± 0.42	**76.61 ± 0.46**
AGPF-FSFG (PR 22')[17]	78.54 ± 0.83	77.92 ± 0.94	82.65 ± 0.89	83.94 ± 0.76	72.06 ± 0.91
LCCRN (TCSVT 23')[29]	82.97 ± 0.19	82.39 ± 0.19	88.48 ± 0.17	87.04 ± 0.17	75.87 ± 0.20
Ours	<u>83.82 ± 0.18</u>	**83.51 ± 0.19**	**88.94 ± 0.16**	**88.02 ± 0.16**	<u>76.54 ± 0.21</u>
	5-Way 5-Shot Accuracy (%)				
MatchingNet (NeurIPS 16')[20]	85.17 ± 0.60	87.61 ± 0.55	88.99 ± 0.50	87.61 ± 0.55	79.57 ± 0.63
ProtoNet (NeurIPS 17')[8]	91.15 ± 0.11	89.46 ± 0.14	93.51 ± 0.09	93.11 ± 0.10	86.54 ± 0.13
RelationNet (CVPR 18')[9]	77.87 ± 0.64	86.84 ± 0.56	86.62 ± 0.55	68.52 ± 0.78	66.20 ± 0.74
Baseline++ (CVPR 19')[1]	81.15 ± 0.61	85.72 ± 0.63	88.06 ± 0.44	84.17 ± 0.58	77.36 ± 0.62
DeepEMD (CVPR 20')[12]	86.30 ± 0.19	83.63 ± 0.26	85.17 ± 0.28	88.37 ± 0.17	81.13 ± 0.20
VFD (ICCV 21')[35]	91.48 ± 0.39	89.90 ± 0.53	88.77 ± 0.46	90.76 ± 0.46	88.00 ± 0.47
FRN (CVPR 21')[13]	92.59 ± 0.10	92.52 ± 0.11	94.21 ± 0.08	94.78 ± 0.08	88.22 ± 0.12
TDM (CVPR 22')[30]	92.37 ± 0.10	93.60 ± 0.10	94.47 ± 0.08	96.11 ± 0.07	88.32 ± 0.12
MCL-Katz (CVPR 22')[16]	93.18 ± 0.00	90.31 ± 0.00	93.28 ± 0.00	93.92 ± 0.00	85.24 ± 0.00
DeepBDC (CVPR 22')[25]	92.10 ± 0.24	92.82 ± 0.24	93.25 ± 0.18	94.97 ± 0.19	**90.22 ± 0.23**
AGPF-FSFG (PR 22')[17]	89.85 ± 0.44	91.96 ± 0.45	89.25 ± 0.49	94.11 ± 0.36	84.83 ± 0.50
LCCRN (TCSVT 23')[29]	**93.63 ± 0.10**	94.24 ± 0.09	94.61 ± 0.08	96.19 ± 0.07	88.36 ± 0.11
Ours	<u>93.45 ± 0.10</u>	**95.27 ± 0.08**	**94.88 ± 0.08**	**96.23 ± 0.07**	<u>88.52 ± 0.12</u>

opposite effect to the CE loss and can prevent the learned model from being over-confident in its predictions on the training set:

$$L_{RCE} = -\frac{1}{m}\sum_{q=1}^{m}\left(\left(\mathbf{1}-\boldsymbol{y}_q\right)^{\top}\log\left(\boldsymbol{p}_q\right)\right). \tag{14}$$

The final loss is obtained by combining the CE loss and RCE loss:

$$\begin{aligned} Loss &= L_{CE} + kL_{RCE} \\ &= -\frac{1}{m}\sum_{q=1}^{m}\left(\left[\boldsymbol{y}_q^{\top} + k\left(\mathbf{1}-\boldsymbol{y}_q\right)^{\top}\right]\log\left(\boldsymbol{p}_q\right)\right), \end{aligned} \tag{15}$$

where $0 < k < 1$ adjusts the influence of the RCE loss. When combined with the CE loss as in Eq. (15), the RCE loss can effectively restrict the decrease of CE loss. Moreover, when $0 < k < 1$, the joint loss trades off between assigning the probability of 1 to the correct class and assigning equal probabilities to all classes. Therefore, the training process will encourage the query sample to be correctly classified as usual, but not necessarily with a high classification probability, thus preventing the model from overfitting to noises or non-generalizable features and improving generalization [34].

4. Experiments and discussions

To evaluate the effectiveness of our proposed approach, we present in this section experiments designed for six purposes:

- To compare our proposed method with state-of-the-art methods for the task of few-shot fine-grained image classification (Section 4.2);
- To study the effectiveness of each branch of our network in image classification and in mitigating the overfitting problem (Sections 4.3, 4.5.1, 4.5.3);
- To investigate the stability of the classification accuracy (Section 4.4);
- To evaluate the discriminative power of the learned features (Sections 4.5.4, 4.5.2);
- To illustrate the effect of feature reconstruction (Section 4.5.5);
- To assess the computational complexity (Section 4.6).

4.1. Implementation details

We conduct experiments on the following five popular benchmark datasets: CUB-200-2011 (CUB) [36], Stanford-Cars (Cars) [37], Stanford-Dogs (Dogs) [38], Flowers [39], and FGVC-Aircraft (Aircraft) [40]. The CUB dataset is consistent with that in [13], and the images are pre-cropped into the bounding boxes provided. All datasets are divided into base, validation, and novel datasets according to the ratio of 2:1:1.

For the feature extractor, we adopt two widely-used backbones: ResNet-12 and ResNet-18 [1,13]. The ResNet-12 structure has 4 residual blocks, and each residual block contains 3 convolutional layers with 3×3 convolution kernel. Each convolutional layer is followed by a batch normalization, and the first convolutional layer is followed by a ReLU nonlinearity, while a 2×2 maximum pooling layer is added at the end of each residual block. The input dimension of the network is $3 \times 84 \times 84$, and the output feature dimension is $640 \times 5 \times 5$ after feature extraction.

Unlike the ResNet-18 network in [41], our ResNet-18 network is modified on the ResNet-12 network. [41] uses a 7×7 convolution kernel in the first convolutional layer, which is not conducive to fine-grained feature extraction as the subtle discriminative features may locate in a tiny region in fine-grained image classification. Our ResNet-18 network, like ResNet-12, has 4 residual blocks, but the first two residual blocks are divided into two sub-residual blocks; each sub-residual block contains 3 convolutional layers with 3×3 convolution kernel. The rest of the structure is similar to ResNet-12.

Throughout the experiments, we use the setting of 10-way 5-shot for training, 5-way 5-shot for validating, and 5-way 1-shot and 5-way 5-shot for testing. The query set contains 16 images in all three phases. In the training phase, we use the SGD optimizer on all datasets with an initial learning rate of 0.1 and momentum of 0.9, and train 1200

Table 2
Comparison of few-shot classification methods on CUB-200–2011, Flowers, FGVC Aircraft, Stanford-Cars, and Stanford-Dogs datasets. All experiments adopt ResNet-18 as the backbone network. Mean accuracy and 95% confidence interval are reported. The best-performing methods are shown in bold and the second best ones are underlined.

Method	CUB	Flowers	Aircraft	Cars	Dogs
	5-Way 1-Shot Accuracy (%)				
MatchingNet (NeurIPS 16')[20]	72.88 ± 0.89	76.07 ± 0.82	82.84 ± 0.81	75.03 ± 0.95	65.59 ± 0.95
RelationNet (CVPR 18')[9]	68.82 ± 1.04	69.04 ± 0.97	74.76 ± 0.97	64.08 ± 1.05	54.21 ± 1.00
Baseline++ (CVPR 19')[1]	65.67 ± 0.95	67.90 ± 0.96	75.92 ± 0.88	67.41 ± 0.99	62.54 ± 0.87
Neg-margin (CVPR 19')[6]	72.51 ± 0.82	76.34 ± 0.89	77.40 ± 0.86	76.04 ± 0.81	68.86 ± 0.83
FRN (CVPR 21')[13]	83.40 ± 0.19	81.22 ± 0.21	87.89 ± 0.18	87.63 ± 0.17	77.53 ± 0.21
TDM (CVPR 22')[30]	83.25 ± 0.19	82.31 ± 0.20	87.91 ± 0.17	87.69 ± 0.17	76.59 ± 0.21
MCL-Katz (CVPR 22')[16]	**85.84 ± 0.00**	76.57 ± 0.00	88.44 ± 0.00	86.12 ± 0.00	72.07 ± 0.00
DeepBDC (CVPR 22')[25]	81.85 ± 0.42	81.07 ± 0.50	85.22 ± 0.41	85.48 ± 0.40	**78.81 ± 0.43**
AGPF-FSFG (PR 22')[17]	79.02 ± 0.83	78.69 ± 0.84	85.02 ± 0.86	84.68 ± 0.78	73.61 ± 0.91
LCCRN (TCSVT 23')[29]	82.80 ± 0.19	82.86 ± 0.19	88.66 ± 0.18	86.24 ± 0.18	77.29 ± 0.20
Ours	84.14 ± 0.18	**83.25 ± 0.19**	**89.14 ± 0.17**	**88.70 ± 0.16**	77.57 ± 0.20
	5-Way 5-Shot Accuracy (%)				
MatchingNet (NeurIPS 16')[20]	85.25 ± 0.57	87.46 ± 0.51	88.77 ± 0.54	87.02 ± 0.56	80.94 ± 0.60
RelationNet (CVPR 18')[9]	82.68 ± 0.58	85.46 ± 0.58	87.45 ± 0.55	91.45 ± 0.44	80.42 ± 0.62
Baseline++ (CVPR 19')[1]	81.53 ± 0.58	84.34 ± 0.62	88.13 ± 0.47	85.50 ± 0.58	79.04 ± 0.61
Neg-margin (CVPR 19')[6]	89.25 ± 0.43	90.83 ± 0.47	90.92 ± 0.39	93.06 ± 0.38	85.75 ± 0.52
FRN (CVPR 21')[13]	92.69 ± 0.10	92.33 ± 0.11	93.96 ± 0.09	95.35 ± 0.08	89.05 ± 0.11
TDM (CVPR 22')[30]	92.98 ± 0.10	93.46 ± 0.11	94.28 ± 0.08	96.06 ± 0.07	88.87 ± 0.11
MCL-Katz (CVPR 22')[16]	93.29 ± 0.00	90.06 ± 0.00	93.22 ± 0.00	93.35 ± 0.00	85.46 ± 0.00
DeepBDC (CVPR 22')[25]	93.00 ± 0.24	93.19 ± 0.24	94.26 ± 0.16	95.84 ± 0.16	**91.33 ± 0.22**
AGPF-FSFG (PR 22')[17]	89.92 ± 0.42	92.78 ± 0.40	91.94 ± 0.47	94.87 ± 0.33	85.68 ± 0.52
LCCRN (TCSVT 23')[29]	**93.60 ± 0.10**	93.87 ± 0.10	94.72 ± 0.07	96.34 ± 0.07	89.54 ± 0.10
Ours	93.58 ± 0.09	**94.70 ± 0.09**	**94.79 ± 0.08**	**96.40 ± 0.07**	89.20 ± 0.11

epochs in total. In the testing phase, the accuracy score is obtained by averaging over 10,000 trials. The coefficient k in Eq. (15) is chosen separately for each dataset according to the accuracy on the validation set. The code of our method is available at https://github.com/liz-lut/SRM-main.

4.2. Comparison with state-of-the-arts

To validate the efficacy of our method, we compare it with the following 13 methods: (1) four classical few-shot learning methods, namely MatchingNet [20], ProtoNet [8], RelationNet [9], Baseline++ [1]; (2) three state-of-the-art metric-based methods, namely DeepEMD [12], MCL-Katz [16], DeepBDC [25]; (3) five state-of-the-art fine-grained few-shot methods, namely VFD [35], FRN [13], TDM [30], AGPF-FSFG [17], LCCRN [29]; and (4) one method that targets the generalizability, namely Neg-margin [6]. Tables 1 and 2 list the performance of these methods with ResNet-12 and ResNet-18 as the backbone, respectively. From the experimental results, we observe that our approach achieves the highest classification accuracy on three datasets for both 5-way 1-shot and 5-way 5-shot classification and is the second-best in most other cases. In particular, compared with FRN which reconstructs query features only from support features, our approach, adding self-reconstruction and restrained cross-entropy loss, shows 2.44% gains in 5-way 1-shot classification and 2.75% gains in the 5-way 5-shot classification on the Flowers dataset with the ResNet-12 backbone; similar improvements can be observed with the ResNet-18 backbone. This demonstrates the effectiveness of our proposed method and encourages the use of self-reconstruction and restrained cross-entropy loss.

4.3. Ablation studies

To further demonstrate the effectiveness of our proposed network, we conduct 5-way 1-shot and 5-way 5-shot experiments on four model structures on Flowers and Cars datasets with ResNet-12 backbone. The four models are as follows: ProtoNet, ProtoNet with our proposed self-reconstruction metric module (SRM), ProtoNet with our proposed loss L_{RCE}, and ProtoNet with both SRM and L_{RCE} (i.e., Ours). As

Table 3
Ablation studies on the Flowers and Cars datasets. All experiments adopt the ResNet-12 backbone. Mean accuracy and its 95% confidence interval are reported. ProtoNet with both SRM and L_{RCE} is our proposed network, noted in the table as Ours.

	Method	Flowers	Cars
5-Way 1-Shot	ProtoNet	75.41 ± 0.22	82.29 ± 0.20
	ProtoNet+L_{RCE}	76.62 ± 0.22	85.25 ± 0.19
	ProtoNet+SRM	82.67 ± 0.19	85.74 ± 0.18
	Ours	**83.51 ± 0.18**	**88.02 ± 0.16**
5-Way 5-Shot	ProtoNet	89.46 ± 0.14	93.11 ± 0.10
	ProtoNet+L_{RCE}	89.18 ± 0.14	93.55 ± 0.10
	ProtoNet+SRM	94.89 ± 0.08	95.37 ± 0.08
	Ours	**95.27 ± 0.08**	**96.23 ± 0.07**

shown in Table 3, when using the SRM module on the ProtoNet, the accuracy is higher than ProtoNet in all cases. When using loss L_{RCE} on ProtoNet, the accuracy is slightly lower than ProtoNet on the 1-shot task on Flowers, while it improves in other cases. Finally, when the SRM module and L_{RCE} are used together (Ours), the accuracy increases dramatically. This highlights the importance of combining both techniques to get a better network.

4.4. Stability analysis

4.4.1. Boxplots of classification accuracy

Tables 1 and 2 list the mean value and 95% confidence interval of test accuracy. To provide a further insight into the classification performance, we present the boxplots of classification accuracy of ProtoNet, FRN, and our proposed method on CUB, Dogs, Flowers, and Cars datasets in Fig. 3. All experiments are based on a 5-way 5-shot classification setup with the ResNet-12 backbone. On each dataset, 100 tasks were randomly selected and used to evaluate the methods.

As can be seen from Fig. 3, our method is obviously better than the other two methods in terms of the median (orange lines) and mean (green dashed lines). Moreover, the range of classification accuracy excluding outliers (i.e., the distance between whiskers) of our method is significantly narrower than that of ProtoNet and FRN, indicating that our method is more stable and has higher confidence. Furthermore,

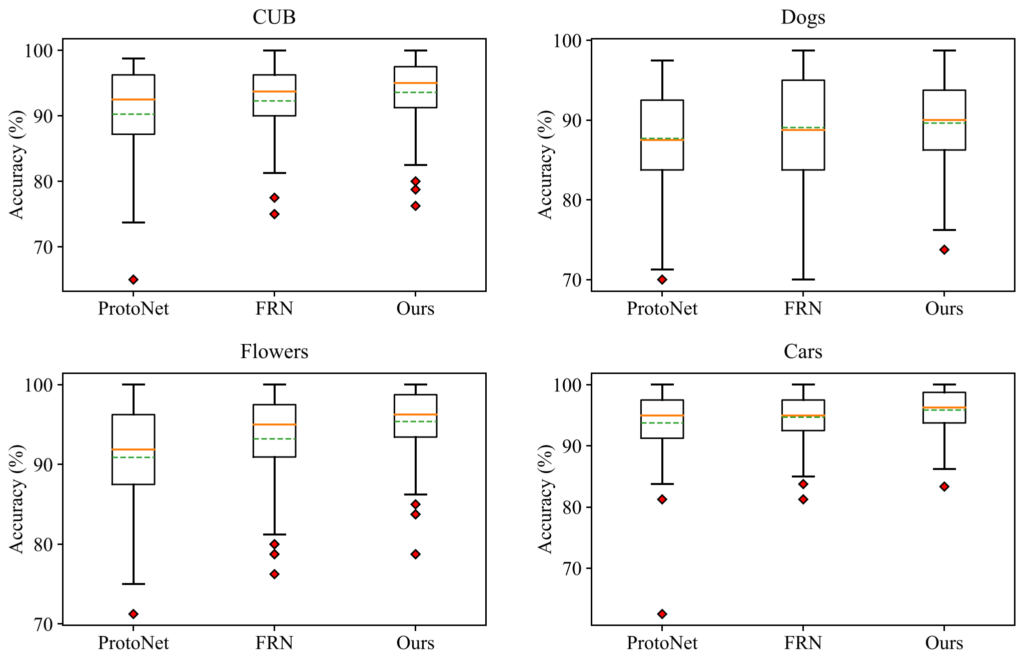


Fig. 3. Boxplots of classification accuracy of ProtoNet, FRN, and our proposed method on CUB, Dogs, Flowers, and Cars datasets. All experiments are based on a 5-way 5-shot classification setup with the ResNet-12 backbone. Each method has been evaluated for 100 rounds, and the distributions of test accuracy are shown via boxplots. In each boxplot, the central orange line marks the median, and the green dashed line marks the mean; the edges of the box are the 25th and 75th percentiles, respectively; and the outliers are marked in red individually.

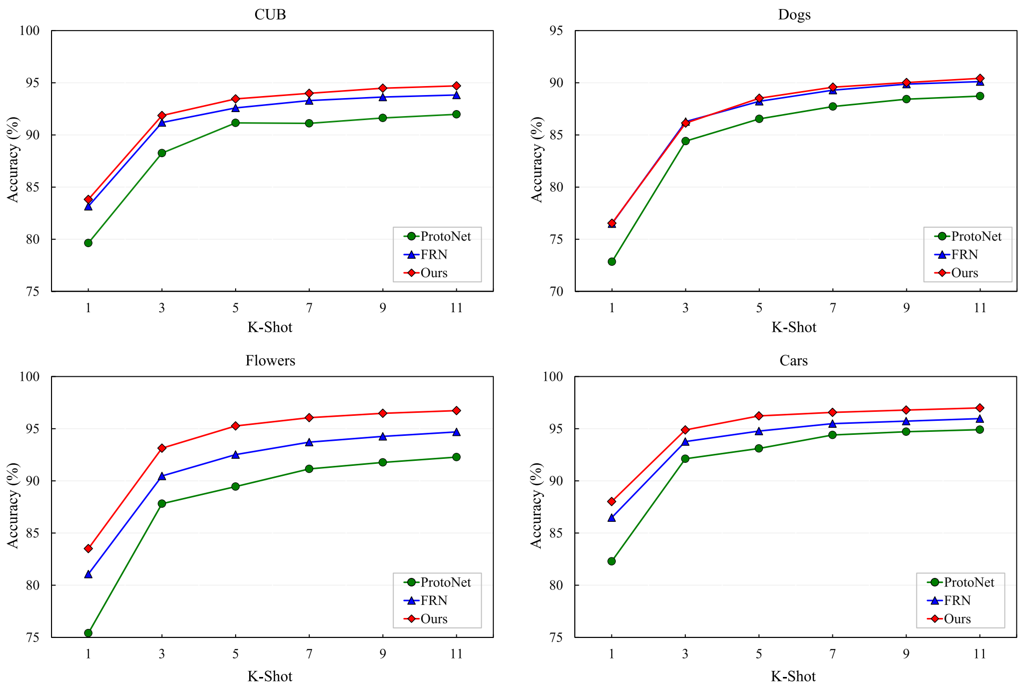


Fig. 4. Test accuracy of ProtoNet, FRN, and our proposed method on CUB, Dogs, Flowers, and Cars datasets in different K-shot settings. All experiments are based on a 5-way classification setup with the ResNet-12 backbone.

looking at the outliers (red points), we can see that the classification accuracy of our method is also better than the other two methods on the worst-performing tasks.

4.4.2. Classification accuracy under different shots

To further evaluate the stability of our method, we calculate the test accuracy of ProtoNet, FRN, and our proposed method in different K-shot settings on CUB, Dogs, Flowers, and Cars datasets. Fig. 4 shows the classification accuracy in different K-shot settings for the 5-way classification task. While our method is only slightly better than ProtoNet and FRN on the Dogs dataset, its advantage is more obvious on other datasets. The advantage over ProtoNet is more pronounced for 1-shot classification, which is reasonable since overfitting is more likely to occur when only one support image is provided. The advantage over FRN holds across different number of shots.

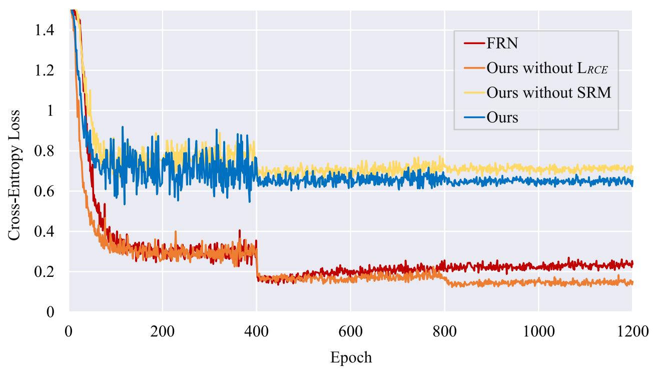


Fig. 5. Validation loss of FRN and our method on the Cars dataset. All experiments are based on a 5-way 5-shot classification setup with the ResNet-12 backbone.

4.5. Visualization analysis

4.5.1. Visualization of validation loss

As discussed in the introduction, FRN suffers from overfitting and, to overcome this issue, we propose both self-reconstruction and restrained cross-entropy loss. Fig. 5 shows the cross-entropy losses of four model structures on Cars; the four models are FRN, ours without the proposed self-reconstruction metric module (SRM), ours without the proposed loss L_{RCE}, and our proposed method. Each data point is calculated on the validation set after each epoch. As we can see, FRN reaches the minimum value of cross-entropy loss at around the 420th epoch and gradually increases afterward, indicating the occurrence of overfitting. However, the cross-entropy loss does not tend to increase after the 400th epoch for our method and its two variants. Therefore, both the proposed SRM structure and restrained cross-entropy loss L_{RCE} can effectively mitigate the overfitting problem.

4.5.2. Visualization of classification probabilities

To further evaluate the effectiveness of the proposed RCE loss in preventing over-confident predictions, we present a heat map of the classification probabilities of query samples predicted by ProtoNet, FRN, and our method (defined by Eq. (12)) on the test set of Cars dataset. As shown in Fig. 6, in each confusion matrix, the vertical axis shows 5 classes in a task, and the horizontal axis shows query samples in the 5 classes with each class containing 16 query samples. The main diagonal line indicates the correct classification. Warmer color represents higher probability score.

Firstly, we notice that ProtoNet and FRN make a number of incorrect predictions for 1-shot classification. Taking class 3 as an example, the probabilities of assigning queries of class 3 to the correct class are often lower than the probabilities of assigning them to other classes, resulting in incorrect predictions. These match the classification accuracy shown below their respective figures. Secondly, we see that our method makes correct predictions on more query samples as the correct class is often assigned with the highest classification probabilities. Moreover, the correct classification probabilities of our method are typically not very high, mostly ranging from 0.4 to 0.7. This is a consequence of including the restricted cross-entropy in the loss function to prevent over-confident predictions, aligning with the findings drawn from Fig. 5.

4.5.3. Visualization of discriminative feature regions

To demonstrate the effectiveness of our proposed method on feature extraction, we show the feature regions extracted from the original image by visualizing them using Grad-CAM [42].

Fig. 7 shows that the discriminative feature regions extracted by our method are more concentrated compared with ProtoNet and FRN. It also extracts discriminative features that ProtoNet and FRN do not capture well, so our method could generalize better and become more robust in the presence of noises in some spatial regions.

Table 4
Comparison of model complexity and computational cost.

Method	Number of parameters	Training time (per epoch)		
		CUB	Flowers	Cars
FRN	12,424,323	11.96 s	12.34 s	21.03 s
Ours	12,424,325	12.45 s	12.62 s	22.23 s

4.5.4. Visualization of feature embeddings

We also visualize the feature embeddings learned by ProtoNet, FRN, and our method by t-distributed stochastic neighbor embedding (t-SNE) [43] on the Flowers dataset and show the results in Fig. 8. As can be seen from the figure, the distribution of feature embeddings of our method is relatively concentrated, and the inter-class margin is large. The two variants of our method, ours without self-reconstruction metric module (SRM) or loss L_{RCE}, also show larger inter-class margin compared with FRN and ProtoNet. Thus, our approach improves separability between classes, which is more favorable for the task of few-shot classification.

4.5.5. Visualization of reconstructed features

To provide a deeper insight into the reconstruction module, we train an image generator to recover images from query features. Specifically, three types of query features are considered, namely original features obtained directly after the feature extraction module, self-reconstructed query features, and support-reconstructed query features. To map features back to images, an inverted ResNet-12 decoder is trained on the 5-way 5-shot task to decode features of dimension 640 × 5 × 5 into 3 × 84 × 84. We use the Adam optimizer and L_1 loss to train the decoder. The initial learning rate of the optimizer is set as 0.01, and the batch size is set as 200. After training for 2000 epochs, we select the parameter with the minimum loss as the parameter of the decoder.

In Fig. 9, panel (a) shows 5 query images of the same class, and panel (e) shows 5 support images from 5 classes, where images in the third column has the same class as the query. Panels (b)–(d) are images generated from query features after feature extraction, i.e., Q_q, from self-reconstructed query features, i.e., $\bar{Q}_q$, and from support-reconstructed query features, i.e., $\bar{Q}_c$. The (i,j)th image in panel (d) is the recovered image for ith query by using query features reconstructed from the jth support class.

As we can see from Fig. 9, firstly, images recovered from self-reconstructed query features (panel (c)) are very similar to those recovered from the original query features (panel (b)), but there are some differences. For example, in the second image, feather is more blurred in the self-reconstructed case than in the original case; in the fifth image, beak is missing in the self-reconstructed case. This suggests that the self-reconstructed query features increase the feature diversity, especially generating some hard samples for classification, and therefore help alleviate overfitting. Secondly, as seen from panel (d), images generated from the query features that were reconstructed from same-class support features (i.e., the middle column in that panel) are much more similar to the ground-truth query images than those based on different-class reconstructions (i.e., other four columns in that panel). This shows that our proposed reconstruction module preserves the merit of FRN in reconstructing semantically faithful images from support images.

4.6. Computational cost

We evaluate the computational complexity of the proposed method. Table 4 lists the model complexity in terms of the number of parameters and the computational cost in terms of the training time per epoch. All methods were implemented by using PyTorch on an NVIDIA RTX 3090 GPU. Compared with FRN, our method introduces an additional self-reconstruction step and distance calculation. The self-reconstruction step only adds two new parameters, namely ρ and λ in Eq. (6). Moreover, the increase in training time is marginal.

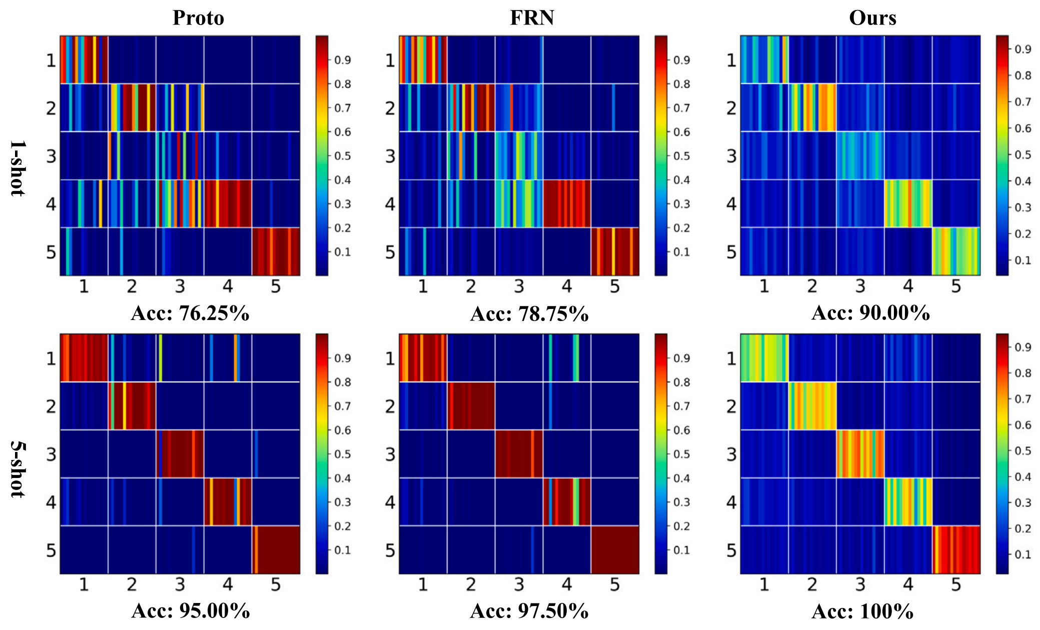


Fig. 6. Visualization of classification probabilities predicted by ProtoNet, FRN and our proposed method on the test set of Cars dataset. In each confusion matrix, the vertical axis shows 5 classes in a task and the horizontal axis shows query samples in the 5 classes with each class containing 16 query samples. Warmer color means higher probability. Classification accuracy of each method is displayed below its respective figure.

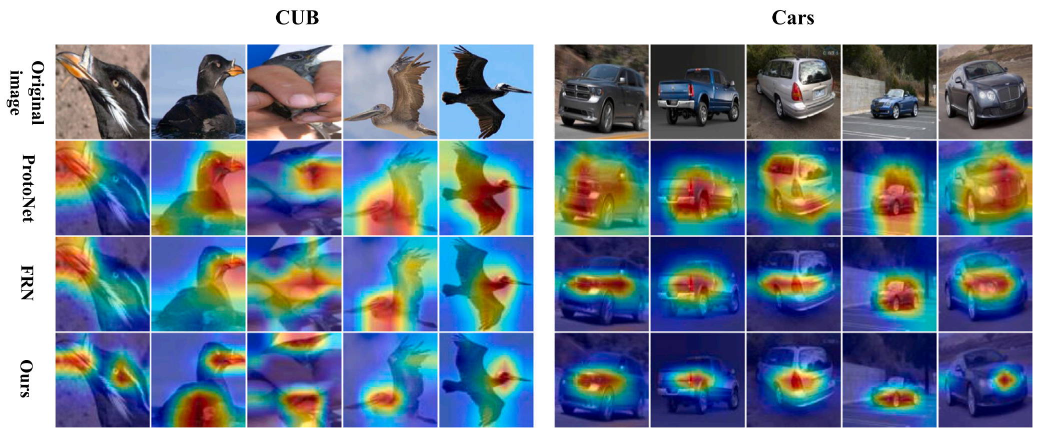


Fig. 7. Feature visualization under ProtoNet, FRN, and our method on the CUB (left) and Cars (right) datasets. Red indicates the learned discriminative features. The redder the region, the more discriminative the learned features.

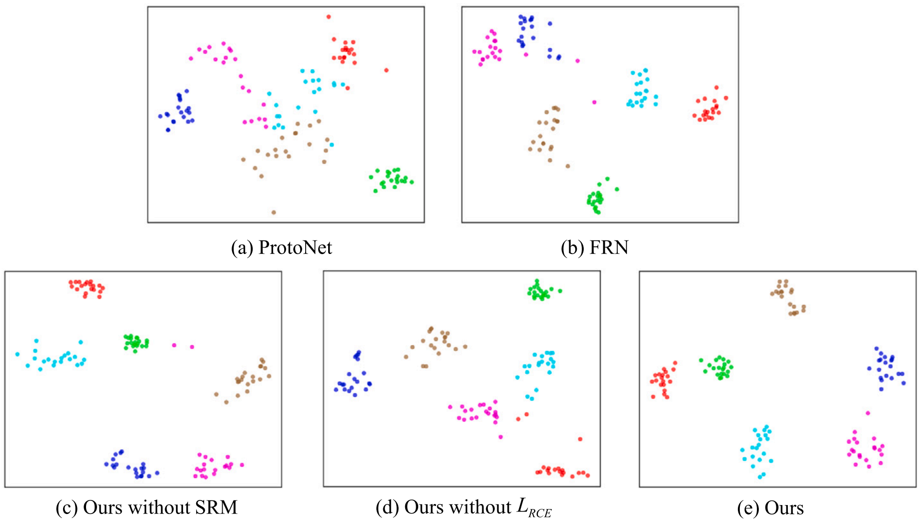


Fig. 8. The t-SNE visualization of convolutional feature of samples by ProtoNet, FRN, and Ours with the ResNet-12 backbone on the Flowers dataset.

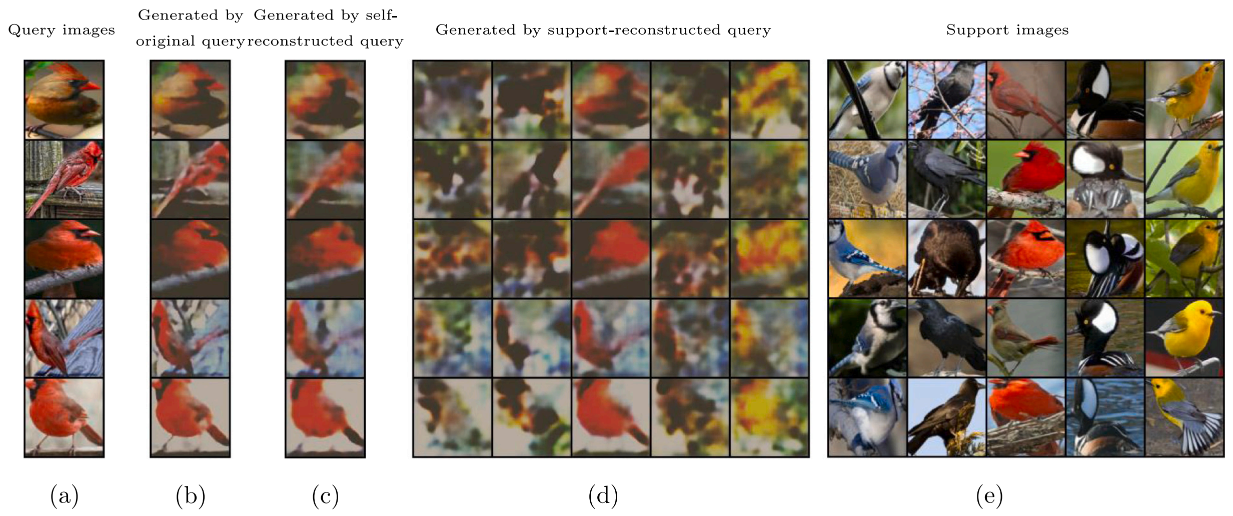

Query images
Generated by original query
Generated by self-reconstructed query
Generated by support-reconstructed query
Support images
(a)
(b)
(c)
(d)
(e)

Fig. 9. Visualization of image reconstruction for CUB. Panel (a) shows 5 query images. Panel (e) shows 25 support images (5 in each class on a column). Panels (b)–(d) are images generated by using original features after feature extraction, self-reconstructed query features, and support-reconstructed query features, respectively. Self-reconstruction increases the diversity, especially the hard samples, and reconstruction from the same class is more credible than that from different classes.

5. Conclusion

In this paper, we proposed a self-reconstruction network for few-shot fine-grained image classification. Our innovation includes enhancing feature diversity by self-reconstructing query samples and introducing restrained cross-entropy loss to mitigate overfitting. Extensive experiments on five benchmark fine-grained datasets demonstrate the efficacy of our method with the state-of-the-art performance achieved on both 5-way 1-shot and 5-way 5-shot classification tasks.

We can observe in Fig. 4 a big performance increase from 1-shot classification to 3-shot classification. This is likely due to the lack of important information in the support images in the 1-shot situation. If we can simulate support images by using some generative models or expand the support feature set by dynamically utilizing query images in a semi-supervised way, the accuracy of 1-shot classification may improve substantially.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data studied in the manuscript are publicly available. Link to code is included in the manuscript.

Acknowledgments

This work was supported in part by the Beijing Natural Science Foundation, China Project No. Z200002, in part by the Royal Society, United Kingdom under International Exchanges Award IEC\NSFC\201071, in part by the National Natural Science Foundation of China (NSFC) No. 62111530146, 62176110, 61906080, 61922015, U19B2036, 62225601, in part by Young Doctoral Fund of Education Department of Gansu Province, China under Grant 2021QB-038, Youth Innovative Research Team of BUPT, China No. 2023QNTD02, and Hong-liu Distinguished Young Talents Foundation of Lanzhou University of Technology, China.

References

[1] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C.F. Wang, J.-B. Huang, A closer look at few-shot classification, in: International Conference on Learning Representations, 2018.
[2] Y. Wang, Q. Yao, J.T. Kwok, L.M. Ni, Generalizing from a few examples: A survey on few-shot learning, ACM Comput. Surv. (csur) 53 (3) (2020) 1–34.
[3] Y. Guo, R. Du, X. Li, J. Xie, Z. Ma, Y. Dong, Learning calibrated class centers for few-shot classification by pair-wise similarity, IEEE Trans. Image Process. 31 (2022) 4543–4555.
[4] X. Li, X. Yang, Z. Ma, J.-H. Xue, Deep metric learning for few-shot image classification: A review of recent developments, Pattern Recognit. (2023) 109381.
[5] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 1126–1135.
[6] B. Liu, Y. Cao, Y. Lin, Q. Li, Z. Zhang, M. Long, H. Hu, Negative margin matters: Understanding margin in few-shot classification, in: European Conference on Computer Vision, 2020, pp. 438–455.
[7] W. Chen, Z. Zhang, W. Wang, L. Wang, Z. Wang, T. Tan, Few-shot learning with unsupervised part discovery and part-aligned similarity, Pattern Recognit. 133 (2023) 108986.
[8] J. Snell, K. Swersky, R.S. Zemel, Prototypical networks for few-shot learning, Adv. Neural Inf. Process. Syst. 30 (2017).
[9] F. Sung, Y. Yang, L. Zhang, T. Xiang, P.H.S. Torr, T.M. Hospedales, Learning to compare: Relation network for few-shot learning, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 1199–1208.
[10] W. Li, L. Wang, J. Xu, J. Huo, Y. Gao, J. Luo, Revisiting local descriptor based image-to-class measure for few-shot learning, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 7253–7260.
[11] Z. Wu, Y. Li, L. Guo, K. Jia, PARN: Position-aware relation networks for few-shot learning, in: IEEE/CVF International Conference on Computer Vision, 2019, pp. 6658–6666.
[12] C. Zhang, Y. Cai, G. Lin, C. Shen, DeepEMD: Few-shot image classification with differentiable earth mover's distance and structured classifiers, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12200–12210.
[13] D. Wertheimer, L. Tang, B. Hariharan, Few-shot classification with feature map reconstruction networks, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8008–8017.
[14] Y. Liu, Y. Bai, X. Che, J. He, Few-shot fine-grained image classification: A survey, in: International Conference on Natural Language Processing, IEEE, 2022, pp. 201–211.
[15] Y. Yu, D. Zhang, S. Wang, Z. Ji, Z. Zhang, Local spatial alignment network for few-shot learning, Neurocomputing 497 (2022) 182–190.
[16] Y. Liu, W. Zhang, C. Xiang, T. Zheng, D. Cai, X. He, Learning to affiliate: Mutual centralized learning for few-shot classification, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 14391–14400.
[17] H. Tang, C. Yuan, Z. Li, J. Tang, Learning attention-guided pyramidal features for few-shot fine-grained recognition, Pattern Recognit. 130 (2022) 108792.
[18] Y.-X. Wang, R. Girshick, M. Hebert, B. Hariharan, Low-shot learning from imaginary data, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7278–7286.

[19] C. Wang, S. Song, Q. Yang, X. Li, G. Huang, Fine-grained few shot learning with foreground object transformation, Neurocomputing 466 (2021) 16–26.
[20] O. Vinyals, C. Blundell, T.P. Lillicrap, K. Kavukcuoglu, D. Wierstra, Matching networks for one shot learning, in: Advances in Neural Information Processing Systems, 2016.
[21] H. Huang, Z. Wu, W. Li, J. Huo, Y. Gao, Local descriptor-based multi-prototype network for few-shot learning, Pattern Recognit. 116 (2021) 107935.
[22] K. Cao, M. Brbic, J. Leskovec, Concept learners for few-shot learning, in: International Conference on Learning Representation, 2021.
[23] Y. Zhou, Y. Guo, S. Hao, R. Hong, Hierarchical prototype refinement with progressive inter-categorical discrimination maximization for few-shot learning, IEEE Trans. Image Process. 31 (2022) 3414–3429.
[24] X. Huang, S.H. Choi, SAPENet: self-attention based prototype enhancement network for few-shot learning, Pattern Recognit. 135 (2023) 109170.
[25] J. Xie, F. Long, J. Lv, Q. Wang, P. Li, Joint distribution matters: Deep Brownian distance covariance for few-shot classification, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 7972–7981.
[26] X. Li, J. Wu, Z. Sun, Z. Ma, J. Cao, J.-H. Xue, BSNet: Bi-similarity network for few-shot fine-grained image classification, IEEE Trans. Image Process. 30 (2020) 1318–1331.
[27] W. Zhu, W. Li, H. Liao, J. Luo, Temperature network for few-shot learning with distribution-aware large-margin metric, Pattern Recognit. 112 (2021) 107797.
[28] H. Huang, J. Zhang, J. Zhang, J. Xu, Q. Wu, Low-rank pairwise alignment bilinear network for few-shot fine-grained image classification, IEEE Trans. Multimed. 23 (2021) 1666–1680.
[29] X. Li, Q. Song, J. Wu, R. Zhu, Z. Ma, J.-H. Xue, Locally-enriched cross-reconstruction for few-shot fine-grained image classification, IEEE Trans. Circuits Syst. Video Technol. (2023) 1.
[30] S. Lee, W. Moon, J.-P. Heo, Task discrepancy maximization for fine-grained few-shot classification, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 5321–5330.
[31] Z. Li, Z. Hu, W. Luo, X. Hu, SaberNet: Self-attention based effective relation network for few-shot learning, Pattern Recognit. 133 (2023) 109024.
[32] B. Munjal, A. Flaborea, S. Amin, F. Tombari, F. Galasso, Query-guided networks for few-shot fine-grained classification and person search, Pattern Recognit. 133 (2023) 109049.
[33] K.B. Petersen, M.S. Pedersen, et al., The matrix cookbook, Tech. Univ. Denmark 7 (15) (2008) 510.
[34] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, G. Hinton, Regularizing neural networks by penalizing confident output distributions, in: International Conference on Learning Representations, 2017.
[35] J. Xu, H. Le, M. Huang, S. Athar, D. Samaras, Variational feature disentangling for fine-grained few-shot classification, in: IEEE/CVF International Conference on Computer Vision, 2021, pp. 8812–8821.
[36] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200–2011 Dataset, California Institute of Technology, 2011.
[37] J. Krause, M. Stark, J. Deng, L. Fei-Fei, 3D object representations for fine-grained categorization, in: IEEE International Conference on Computer Vision Workshops, 2013, pp. 554–561.
[38] A. Khosla, N. Jayadevaprakash, B. Yao, F.-F. Li, Novel dataset for fine-grained image categorization: Stanford dogs, in: CVPR Workshop on Fine-Grained Visual Categorization, FGVC, 2011.
[39] M.-E. Nilsback, A. Zisserman, Automated flower classification over a large number of classes, in: Indian Conference on Computer Vision, Graphics & Image Processing, 2008, pp. 722–729.
[40] S. Maji, E. Rahtu, J. Kannala, M.B. Blaschko, A. Vedaldi, Fine-grained visual classification of aircraft, 2013, ArXiv, arXiv:1306.5151.
[41] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[42] R.R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, Int. J. Comput. Vis. 128 (2017) 336–359.
[43] L. van der Maaten, G.E. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.

Xiaoxu Li received her Ph.D. degree from Beijing University of Posts and Telecommunications in 2012. She is a Professor with the School of Computer and Communication, Lanzhou University of Technology. Her research interests include machine learning fundamentals with a focus on applications in image and video understanding.

Zhen Li received his B.Eng. degree in Measurement and Control Technology and Instrument from Taiyuan University of Technology, China, in 2012. He is currently pursuing the master's degree at Lanzhou University of Technology. His research interests include computer vision and machine learning.

Jiyang Xie received his B.E. degree in information engineering from Beijing University of Posts and Telecommunications (BUPT), China, in 2017, where he received his Ph.D. degree in 2022. His research interests include pattern recognition and machine learning fundamentals with a focus on applications in image processing, data mining, and deep learning.

Xiaochen Yang received the Ph.D. degree in statistical science from University College London in 2020. She is a Lecturer with the School of Mathematics and Statistics, University of Glasgow. Her research interests include statistical classification, metric learning, and hyperspectral image analysis. She is an Associate Editor of Neurocomputing.

Jing-Hao Xue received the Dr.Eng. degree in signal and information processing from Tsinghua University in 1998 and the Ph.D. degree in statistics from the University of Glasgow in 2008. He is a Professor with the Department of Statistical Science at University College London. His research interests include multivariate and high-dimensional data analysis, statistical pattern recognition, machine learning and image processing. He is an Associate Editor of IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Cybernetics, and IEEE Transactions on Neural Networks and Learning Systems.

Zhanyu Ma is currently a Professor at Beijing University of Posts and Telecommunications, Beijing, China, since 2019. He received the Ph.D. degree in electrical engineering from KTH Royal Institute of Technology, Sweden, in 2011. From 2012 to 2013, he was a Postdoctoral Research Fellow with the School of Electrical Engineering, KTH. He has been an Associate Professor with the Beijing University of Posts and Telecommunications, Beijing, China, from 2014 to 2019. His research interests include pattern recognition and machine learning fundamentals with a focus on applications in computer vision, multimedia signal processing. He is a Senior Member of IEEE.