Pattern Recognition 163 (2025) 111466

Contents lists available at ScienceDirect

Pattern Recognition

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

CDN4: A cross-view Deep Nearest Neighbor Neural Network for fine-grained few-shot classification

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

[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

Dataset link: https://github.com/ShawnTing1/CDN4

Keywords:
Few-shot learning
Fine-grained image classification
Deep neural network
Data augmentation

ABSTRACT

The fine-grained few-shot classification is a challenging task in computer vision, aiming to classify images with subtle and detailed differences given scarce labeled samples. A promising avenue to tackle this challenge is to use spatially local features to densely measure the similarity between query and support samples. Compared with image-level global features, local features contain more low-level information that is rich and transferable across categories. However, methods based on spatially localized features have difficulty distinguishing subtle category differences due to the lack of sample diversity. To address this issue, we propose a novel method called Cross-view Deep Nearest Neighbor Neural Network (CDN4). CDN4 applies a random geometric transformation to augment a different view of support and query samples and subsequently exploits four similarities between the original and transformed views of query local features and those views of support local features. The geometric augmentation increases the diversity between samples of the same class, and the cross-view measurement encourages the model to focus more on discriminative local features for classification through the cross-measurements between the two branches. Extensive experiments validate the superiority of CDN4, which achieves new state-of-the-art results in few-shot classification across various fine-grained benchmarks. Code is available at https://github.com/ShawnTing1/CDN4.

1. Introduction

Few-shot classification [1] is a paradigm that aims at constructing models with high classification performance for novel classes given only a few labeled samples. It has shown promising results in alleviating the limitations of training deep neural networks in low-data regimes. Research attempts in few-shot classification can be roughly categorized into two schools: meta learning-based [2] and metric learning-based [3,4]. Meta learning-based methods accumulate task-agnostic knowledge from learning multiple tasks and generalize it to a few shots of novel classes. Metric learning based methods learn a deep embedding space driven by the similarity metric between the query and support images for classification.

In metric-based methods, previous literature has adopted image-level global features [1,5–7]. However, the use of global features faces challenges in fine-grained image classification, which aims to distinguish classes within a super-category. For this task, due to the subtle differences between classes, it is crucial for the model to capture the fine details of specific regions within the image. Taking bird classification as an example, Field Sparrow and Savannah Sparrow are two species similar in shape. To distinguish them, the key difference lies in the details of their head markings, where the former has a white eye ring and the latter has yellow eyebrows. To focus on such small, localized details, methods extracting spatially local features (LFs) are proposed as they are richer in low-level visual information and provide more transferable information across categories [8]. To fully exploit LFs, recent methods have considered feature representations of semantic patches [9–14]. For example, DN4 [9] uses the local invariant features of two images instead of their image-level representations, and for each LF from a query image, it utilizes the naive Bayes nearest neighbor [15] algorithm to find the nearest neighbor support LFs. It then calculates a similarity score by accumulating all the local scores in a naive Bayes manner, to represent the similarity between the query image and the support class to which the support LFs belong. DN4 is an impressive state-of-the-art method based on LFs with a clear and promising idea and does not introduce complex structures.

Orthogonal to this, many few-shot classification methods [16,17] have adopted the idea of maximizing the consistency of the prediction

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

https://doi.org/10.1016/j.patcog.2025.111466
Received 10 July 2024; Received in revised form 25 January 2025; Accepted 10 February 2025
Available online 18 February 2025

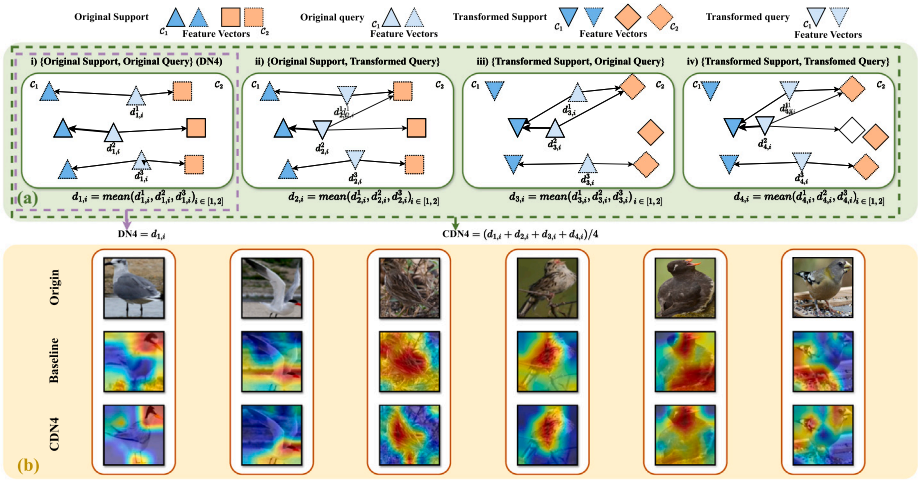

Fig. 1. (a) Comparison between DN4 [9] (baseline) and the proposed CDN4. Each symbol represents a local feature of the image, with color indicating its respective class and a solid or dashed outline indicating the target or the background feature, respectively. The arrows represent the process of finding the nearest supporting local feature for each query local feature. DN4 computes the image-to-class similarity by summing over the similarities between all query local features and their nearest support local features belonging to the ith class, i.e., as in panel (i). CDN4 obtains a different view of support and query images by applying a random geometric transformation, after which the cross-view measurement is conducted, i.e., performing all (i)–(iv). (b) Comparison of the Grad-CAM heatmaps of DN4 and CDN4. The warmer the color in the heat map, the more attention is applied by the model to this region. It can be observed that CDN4 is able to localize the target area better, and the discriminative areas such as the bird head and tail pattern get more attention.

of the query image across different views to make the model focus on discriminative features and different views of the image are obtained by data augmentation. For example, IEPT [16] applies rotation to the image and adds a self-supervised loss to train the network to predict the rotation angle. However, most of them calculate the loss between different augmentations of the same image, which introduces little intra-class variation and thus is not conducive to focusing on discriminative features and improving generalization performance. We argue that augmentations should be applied across samples to increase intra-class variation, and features that remain important under different transformations are more discriminative.

Inspired by the insights discussed above, in this paper, we propose a novel approach termed Cross-view Deep Nearest Neighbor Neural Network (CDN4) for fine-grained few-shot classification, which learns highly discriminative local features under large intra-class variations. Specifically, CDN4 consists of two modules: (I) an Episodic Dual-Branch Structure (EDBS), and (II) a Cross-view Local Metric Module (CLMM). Given an episode, the EDBS treats it as an original episode in the original branch. It also applies a random geometric transformation to obtain a transformed episode in the transformed branch. These two episodes, which can be regarded as two views of the image, are fed into the backbone network to extract feature embedding, resulting in two separate feature branches. In CLMM, we calculate the similarity between local support and query features across the two branches. Compared with conventional approaches which compute the similarity over the original support-original query pair, CLMM incorporates three additional pairs with larger intra-class variation: original support-transformed query, transformed support-original query, and transformed support-transformed query. By pairing original images with different augmented images, we substantially enhance the intra-class variation. Consequently, the network is forced to focus on more discriminative LFs in order to classify the samples correctly under four similarity comparisons. Fig. 1(a) illustrates the difference between CDN4 and the baseline DN4, and Fig. 1(b) illustrates the difference in the heat map between CDN4 and the baseline, from which we can see that CDN4 pays attention to smaller yet highly discriminative regions.

In short, our novelties and contributions are four-fold:

- We propose a novel classification method termed CDN4 to improve the discriminativeness of local features for fine-grained few-shot classification.
- We construct the Episodic Dual-Branch Structure, which adds a new branch with random geometric transformations to introduce more diversity in the samples.
- We construct the Cross-view Local Metrics Module, which focuses on the similarity of four data pairs, forcing the model to learn discriminative features.
- Comprehensive experiments and ablation studies demonstrate that our CDN4 achieves state-of-the-art performance on four few-shot fine-grained image classification benchmarks.

2. Related work

2.1. Metric learning-based few-shot learning

Existing FSL methods can be roughly divided into two main schools: (1) *meta learning-based methods* [2], such as model-agnostic meta-learning (MAML) [18,19], which aim for quick and excellent adaptation to new tasks; (2) *metric learning-based methods* [3], which aim to project samples into a discriminative embedding space and define or learn a metric to measure the similarity between samples. Our work belongs to metric learning-based methods, which can be broadly classified into two streams as below.

Image-level Global Features-based Methods. These methods [1,5–7] represent the entire image using global features from the backbone's last global average pooling layer and treat these features as a data point in the feature space. For example, ProtoNet [5] takes the mean of support samples in each class as the class prototype. It then uses the Euclidean distance to measure the similarity between the query and class-mean features for classification. However, the global representation of an image may overlook local features that are useful for fine-grained image classification.

Spatially Local Features-based Methods. Local features (LFs) provide richer information about different parts of an image, and methods that learn similarities based on LFs have shown improved transferability across categories in FSL [9–14,17,20]. DN4 [9], as one of the classic methods, proposes a local descriptor-based image-to-class measure by using the top-k nearest neighbors to the query features from the pool of support features. Building upon DN4, DMN4 [21] selects task-relevant LFs and discard those from the background by considering the mutual closeness between query and support. FRN [10] reconstructs

query LFs using support LFs and classifies a query instance to the class with the minimum Euclidean distance between the original and reconstructed query LFs. BiFRN [20] additionally reconstructs support LFs using query LFs to reduce intra-class variations. MCL [11] models the bidirectional relationship between support and query LFs using a time-homogeneous Markov process. It computes the random walk probability in both directions, from support to query and from query to support, and makes predictions based on the expected number of visits from the query to support features of the same class. CTNet [13] models long-range spatial dependency by calculating the correlation between pixels and class features, reducing the computational complexity compared with the standard self-attention which computes the correlation between all pixels. More importantly, this method explores the complementarity between the spatial and channel contextual information, as the input for the spatial contextual module consists of middle features and class features, both constructed using multi-scale local channel contexts. Besides adjusting the local importance along the spatial dimension, methods such as TDM [22] adjust channel weights. TDM computes intra-class and inter-class channel-wise representativeness scores and uses them to derive a task-specific weight vector.

In this paper, we build on DN4 and aim to improve the discriminability of LFs. While our work shares a similar motivation as DMN4, the approaches are distinct. DMN4 utilizes a bidirectional indicator to identify discriminative LFs, while we pair up the original support with the transformed query or the transformed support with the original query to enlarges intra-class variation, i.e., forming a more challenging classification task, which facilitates the model to learn more discriminative features.

2.2. Methods related to data augmentation

Methods [23,24] related to data augmentation usually increase the number of training samples and enhance data diversity by applying some transformations, such as cropping, horizontal flipping, translation, or more complex form, to the original image. Other methods apply data augmentation to obtain different views of an image, and then optimize the model to focus on discriminative features by maximizing the similarity between different views. For example, [25] introduces contrast learning to few-shot learning through data augmentation. BANs-SLA [26] uses data augmentation to build self-supervised tasks and predict transformed classes to reduce the generalization error between the base class and the novel class. IEPT [16] generates an augmented image for each image via rotation and computes an auxiliary loss for rotation prediction. Following IEPT, ESPT [17] uses ridge regression to get the relationship between the support image and the query image and again uses data augmentation to compute an auxiliary loss based on the prediction agreement between the two branches. However, these methods all focus on the relationship between an image and its corresponding transformed image, whose difference is relatively limited. Fang et al. [27] point out that augmenting data at the patch level or part level, such as dividing images into patches or replacing image regions with others, may undermine the semantic integrity crucial for fine-grained classification. To address this, a CSDNet is proposed, which generates augmented data that consider the semantic information and amplify the crucial features. It also employs a memory bank to store historical features, thereby expanding samples at the feature level. In addition to mining more information from images, some methods have been proposed to exploit label information. For example, KTN [28] uses a graph convolutional network to build a mapping network between semantic knowledge and visual features, and then combines the knowledge-based classifier and the vision-based classifier through weight fusion, which effectively employs semantic features as a prior knowledge to the few-shot classifier.

Our CDN4 applies a random geometric transformation to obtain the transformed branch from the origin branch and performs the cross-view measurement between the two branches, which can be seen as constructing the dual-branch structure using augmentation. The EDBS in CDN4 is similar to ESPT [17] as both methods construct a two-branch network; however, they differ in implementation details and purpose. ESPT transforms the original image to obtain the transformed branch and transforms the features of the original image to obtain the original branch to ensure that the feature maps of the original and transformed branches are spatially aligned. In contrast, CDN4 keeps the original images intact in the original branch and transforms the images to obtain the transformed branch, thereby introducing larger differences between the two branches. In addition, the transformed branch is used to construct an auxiliary loss in ESPT, whereas it is used for performing cross-view measurements in CDN4. The CLMM in CDN4 is similar to cross-view episodic training in [25], but [25] applies a task-specific module to align features across views, whereas our method does not include such a module since we aim to extract features that are truly discriminative in the presence of larger intra-sample variation.

3. Methodology

3.1. Problem formulation

This work adopts the standard framework of the fine-grained few-shot classification problem. We are given the complete dataset $D = \{(x_i, y_i),\ y_i \in C\}$, where the x_i represents the ith original sample and the y_i represents the sample's class label. The label set C consists of the base categories C_b and the novel categories C_n, and $C_b \cap C_n = \emptyset$. The dataset D is then divided into two subsets accordingly: the base dataset $D_b = \{(x_i, y_i),\ y_i \in C_b\}$ for meta-training and the novel dataset $D_n = \{(x_i, y_i),\ y_i \in C_n\}$ for meta-testing. During the meta-training phase, a deep neural network is trained on the base dataset to obtain an optimal model with learned parameters. In the meta-testing phase, the learned model is evaluated on the novel dataset, provided with very few labeled samples. The ultimate objective of fine-grained few-shot classification is to learn knowledge with high generalization performance from the base dataset D_b that enables accurate classification on a novel and unseen dataset D_n with limited data.

To simulate the test environment and enhance generalization across tasks, we employ an episodic strategy that is widely used in the field of few-shot learning by carrying out a set of episodes (also referred to as tasks) during the meta-training phase. Following this episodic strategy, the base dataset is divided into episodes. In each episode, N randomly selected classes C_b are chosen. Each class consists of K support and q query images, i.e., N-way K-shot episode. Specifically, each episode is denoted as $e = \{S, Q\}$, consisting of a support set $S = \{(x_i, y_i)\}_{i=1}^{N\times K}$ and a query set $Q = \{(x_j, y_j)\}_{j=1}^{N\times q}$. The support and query sets are drawn from the same label space C_b and they do not overlap. The samples in Q are classified based on the samples in S during the episodic training procedure to get the learned model.

In the meta-testing phase, the model selected from the meta-training phase performs N-way K-shot classification on the novel classes C_n. For evaluation purposes, episodes $\mathcal{E}_n$ are typically constructed, and the average accuracy over these episodes is reported.

3.2. Overview of CDN4

The architecture is illustrated in Fig. 2, which consists of an Episodic Dual-Branch Structure (EDBS) f_γ and a Cross-View Local Metric module (CLMM) g_ϕ.

Firstly, in the EDBS f_γ, a random geometric transformation is applied to the original episode to perform image transformation, and the corresponding transformed episode is obtained. Then, features of the original and transformed episodes are extracted through the backbone network f_θ. Finally, the original features and transformed features are input into the CLMM g_ϕ, and the cross-view measurement are constructed in this module to compute the final distance metric.

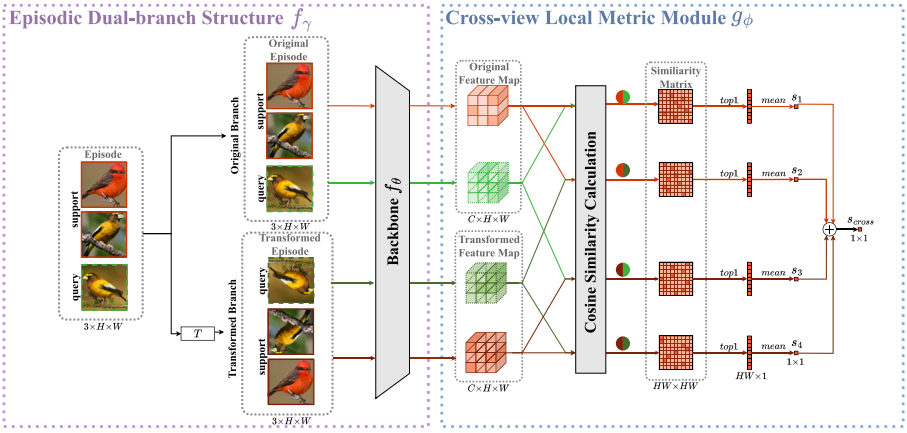

Fig. 2. Architecture of the proposed Cross-view Deep Nearest Neighbor Neural Network (CDN4) for a 2-way 1-shot task. The input episode e, consisting of a support set $\mathcal{S}$ and a query set $\mathcal{Q}$, starts by building an *Episodic Dual-Branch Structure* to get the original branch that contains the original episode e and the transformed branch that contains transformed episode e^T, which are then fed into the backbone to extract original feature map and transformed feature map. In the *Cross-view Local Metric Module*, four similarity matrices are obtained between the original and transformed feature maps of support and query samples, based on which four similarities are calculated by following DN4. The final similarity is obtained through simple averaging.

3.3. Episodic Dual-Branch Structure (EDBS)

In our method, given an episode e, we regard it as an original episode and treat it via the original branch. The transformed branch is obtained by transforming the original episode. These two branches together form the Episodic Dual-Branch Structure.

Precisely, given an original episode $e = \{\mathcal{S}, \mathcal{Q}\}$ containing both the original support and query sets, we randomly rotate all the images in this episode using a random geometric transformation $T \in \{t_r | r = 1, \dots, 3\}$ to get a transformed episode $e^T = \{\mathcal{S}^T, \mathcal{Q}^T\}$, where t_r denotes the operator of rotating the image by $r \times 90°$. In this way, we construct a double branch of episode e and e^T, which we can think of as two separate branches containing the original support set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N\times K}$ and the original query set $\mathcal{Q} = \{(x_j, y_j)\}_{j=1}^{N\times q}$, and the transformed support set $\mathcal{S}^T = \{(T(x_i), y_i)\}_{i=1}^{N\times K}$ and the transformed query set $\mathcal{Q}^T = \{(T(x_j), y_j)\}_{j=1}^{N\times q}$, respectively. Note that this random geometric transformation changes the samples but does not change their labels.

Next, these two episodes are fed into the same backbone network f_θ to get the features $z \in \mathbb{R}^{C\times H\times W}$ to form the support and query sets: $\mathcal{Z}_\mathcal{S} = f_\theta(\mathcal{S}) = \{(z_i, y_i)\}_{i=1}^{N\times K}$ and $\mathcal{Z}_\mathcal{Q} = f_\theta(\mathcal{Q}) = \{(z_j, y_j)\}_{j=1}^{N\times q}$ from the original episode, and $\mathcal{Z}_\mathcal{S}^T = f_\theta(\mathcal{S}^T) = \{(z_i^T, y_i)\}_{i=1}^{N\times K}$ and $\mathcal{Z}_\mathcal{Q}^T = f_\theta(\mathcal{Q}^T) = \{(z_j^T, y_j)\}_{j=1}^{N\times q}$ from the transformed episode. The embedded features z are then transformed into the spatially local features (LFs) $Z \in \mathbb{R}^{C\times HW}$, that is, each image has HW C-dimensional LFs.

It is hoped that LFs can provide different discriminative and transferable information to improve the model's generalization performance. This is an essential clue for image classification in fine-grained few-shot scenarios. At the same time, the transformation does not change the label of the feature, and this dual-branch structure can introduce more sample diversity. In the next subsection, we will calculate the similarity between local features of support and query samples in both their original and transformed views. By maximizing the classification accuracy across all views, we encourage the model to focus on discriminative features.

3.4. Cross-view Local Metric Module (CLMM)

3.4.1. Construct cross-view measurement

By using the above EDBS, we obtain two branches e and e^T, where the original branch contains the original support set $\mathcal{Z}_\mathcal{S}$ and the original query set $\mathcal{Z}_\mathcal{Q}$, and the transformed branch includes the transformed support set $\mathcal{Z}_\mathcal{S}^T$ and the transformed query set $\mathcal{Z}_\mathcal{Q}^T$. After that, four cross-view measurements are constructed: original support-original query pair $\{Z_i, Z_j\}$, original support-transformed query pair $\{Z_i, Z_j^T\}$, transformed support-original query pair $\{Z_i^T, Z_j\}$ and transformed support-transformed query pair $\{Z_i^T, Z_j^T\}$, where $i \in [1, \dots, N]$ for the support samples of the ith class and $j \in [1, \dots, N \times q]$ for the jth query sample to be classified. Note that when $K > 1$, we calculate the prototype of a support class as the representative of that class according to ProtoNet [5].

3.4.2. Local measurement analogously to DN4

After that, we compute the similarities between these individual pairs by adopting the image-to-class measure developed in DN4 [9]. To illustrate the measure, we temporarily omit the superscript T in Z and do not differentiate between the original and transformed data.

Recall that each support and query feature, $Z_i, Z_j \in \mathbb{R}^{C\times m}$, contains $m = H \times W$ LFs. To compute the image-to-class similarity, we find the top-1 nearest neighbor LF from the support sample for each query sample's LF, calculate the similarity between these two LFs, and average over the similarities of all query LFs. More specifically, we first compute the similarity matrix between the ith class and the jth query by using the cosine similarity:

$$M_{ij} = \cos(Z_i, Z_j) = \frac{(Z_j)^\top Z_i}{\|Z_i\| \cdot \|Z_j\|}, \tag{1}$$

where $\cos(\cdot, \cdot)$ represents the cosine similarity, and $M_{ij} \in \mathbb{R}^{HW\times HW}$. Each row of M_{ij} represents the cosine similarity between an LF of the query sample and all the LFs of the support samples of a particular class. Next, for each similarity matrix, we extract the top-1 values of each row to determine the nearest neighbor of each LF of the query sample:

$$M'_{ij} = top(M_{ij}), M'_{ij} \in \mathbb{R}^{HW\times 1}. \tag{2}$$

We note that DN4 uses top-k to find the k nearest neighbors of LFs and then averages the k similarities, while this work uses top-1 for simplicity. Finally, we average over all LFs of the query, which yields the similarity s_{ij} between the ith support class and the jth query image:

$$s_{ij} = mean(M'_{ij}),\ s_{ij} \in \mathbb{R}^{1\times 1}. \tag{3}$$

In sum, the image-to-class measure is computed as follows:

$$s_{ij} = sim(Z_i, Z_j) := mean(top(\cos(Z_i, Z_j))). \tag{4}$$

3.4.3. Similarity measure

For an N-way K-shot episode, the above EDBS generates the original support features Z_i, transformed support features Z_i^T, original query features Z_j, and transformed query features Z_j^T. We can now measure the similarity for all cross-view measurement by computing the aforementioned image-to-class measure.

First, we compute the similarities between the four cross-view data pairs:

$$s_{1,ij} = sim(Z_i, Z_j), \tag{5}$$

$$s_{2,ij} = sim(Z_i^T, Z_j), \tag{6}$$

$$s_{3,ij} = sim(Z_i, Z_j^T), \tag{7}$$

$$s_{4,ij} = sim(Z_i^T, Z_j^T), \tag{8}$$

where $s_{1,ij}$, $s_{2,ij}$, $s_{3,ij}$ and $s_{4,ij}$ indicate the similarities of different pairs between the jth query sample and the ith class.

Following [10,29], we normalize the similarity as

$$\hat{s}_{p,ij} = \frac{\exp(s_{p,ij})}{\sum_{i=1}^{N} \exp(s_{p,ij})}, \ p = 1, 2, 3, 4. \tag{9}$$

3.4.4. Loss functions

Based on the normalized similarities, the cross-entropy loss in an N-way K-shot episode can be calculated as

$$\mathcal{L}_p = -\frac{1}{N \times q} \sum_{j=1}^{N \times q} \sum_{i=1}^{N} \mathbb{1}(y_j == i) \log(\hat{s}_{p,ij}), \ p = 1, 2, 3, 4. \tag{10}$$

Finally, we calculate the simple average of these four loss functions to obtain the cross-view loss $\mathcal{L}_{cross}$:

$$\mathcal{L}_{cross} = \frac{\mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 + \mathcal{L}_4}{4}. \tag{11}$$

In the training phase, the model is based on the cross-view loss function and we use the SGD optimizer on the CUB-200-2011 dataset and the Adam optimizer on the meta-iNat and tiered-meta-iNat datasets to optimize the parameters.

3.4.5. Test phase

In the test phase, to classify the query image, we compute the similarities between the four pairs and use the simple average to obtain the final similarity between the query image and the support class:

$$s_{cross,ij} = \frac{s_{1,ij} + s_{2,ij} + s_{3,ij} + s_{4,ij}}{4}. \tag{12}$$

The model assigns the query image to the class with the largest similarity.

3.5. Full CDN4 algorithm

For easy reproducibility, we present the pseudo-algorithm of CDN4 in Algorithm 1.

Algorithm 1 Training procedure of CDN4

Input: The base dataset D_b, the random geometric transformation $\mathcal{T}$, the maximum number of iterations $Iter_{max}$

Output: The learned parameters θ, γ, ϕ

1: Initialize all learnable parameters θ, γ, ϕ;
2: **for** each $iteration \in [1, Iter_{max}]$ **do**
3: Randomly original episode e from D_b;
4: Generate the transformed episode e^T by applying the random geometric transformation $\mathcal{T}$ to the original episode e;
5: Construct cross-view measurement $\{Z_i, Z_j\}$, $\{Z_i, Z_j^T\}$, $\{Z_i^T, Z_j\}$, $\{Z_i^T, Z_j^T\}$;
6: Compute the similarities between the four cross-view data pairs: $s_{1,ij}$, $s_{2,ij}$, $s_{3,ij}$, $s_{4,ij}$;
7: Compute the cross-view loss: $\mathcal{L}_{cross} = (\mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 + \mathcal{L}_4)/4$;
8: Update θ, γ, ϕ to minimize $\mathcal{L}_{cross}$.
9: **end for**

4. Experiments

4.1. Datasets

Our experiments are conducted on three commonly used datasets for fine-grained few-shot image classification: Caltech-UCSD Birds-200-2011 (CUB-200-2011) [30], meta-iNat [31], tiered-meta-iNat [31], and FGVC-Aircraft (Aircraft) [32]. Following FRN [10], the datasets are split into training, validation (for CUB-200-2011 and Aircraft only), and test sets, with details provided in Table 1.

Table 1
The splits of datasets used in this paper, where C_{train}, C_{val}, C_{test} are the numbers of classes in the training, validation, and test sets, and C_{total} is the total number of classes.

Dataset	C_{train}	C_{val}	C_{test}	C_{total}
CUB-200-2011	100	50	50	200
meta-iNat	908	–	227	1135
tiered-meta-iNat	781	–	354	1135
Aircraft	50	25	25	100

CUB-200-2011 is a well-known dataset for fine-grained image classification, which consists of 11,788 images belonging to 200 different bird species. According to [10,33], we split the dataset into three subsets: 100 classes for training, 50 for validation, and another 50 for testing.[1] Additionally, following [8,29], each image was pre-cropped based on human-annotated bounding boxes.

meta-iNat is a realistic fine-grained benchmark focusing on wild animal species. It contains 1,135 animal species spanning 13 sub-categories, each containing between 50 and 1000 images. We adopt the class split proposed by [10], where 908 classes are for training, and 277 are for testing.

tiered-meta-iNat is a variant of the meta-iNat dataset that consists of the same set of images as meta-iNat while presenting a more challenging scenario by introducing a significant domain gap between the training and test categories. We follow the same split provided by [10], where 781 training classes span the super categories of reptiles, birds, *etc.*, and 354 test classes span the super categories of insects, arachnids, *etc*.

Aircraft consists of 100 aircraft model variants, each with 100 images. The dataset is divided into 50 training categories, 25 validation categories, and 25 test categories.

4.2. Implementation details

Backbone Network. To ensure a fair comparison, we conduct experiments by adopting a commonly used backbone network: ResNet-12 [35]. We remove the last global average pooling layer and leverage the feature pyramid structure to get denser features, which are then used as the feature extractor in our model. Following [8–10], the backbone network is pre-trained on the mini-ImageNet dataset before meta-training. More specifically, it is trained for 350 epochs on mini-ImageNet with a batch size of 128, using the SGD optimizer with an initial learning rate of 0.1 and a decay factor of 0.1 at 200 and 300 epochs.

Training Details. We adopt the setup proposed by FRN [10] for ensuring a fair comparison. For the CUB-200-2011 dataset, we utilize the SGD optimizer with a Nesterov momentum of 0.9 and an initial

[1] There is another division of the CUB-200-2011 dataset proposed in [34], with 120/30/50 classes in the training/validation/test set. This division was used by some methods such as Bi-FRN [20]. For fairness of comparison, all methods evaluated in this paper use the 100/50/50 split.

Table 2
Comparison of five-way classification accuracy with state-of-the-art methods on the fine-grained benchmarks CUB-200-2011, meta-iNat, tiered-meta-iNat, and Aircraft dataset. The accuracy of the methods labeled ¤ are quoted from [11], and the results of the methods labeled † are reproduced in this paper; all results on the Aircraft dataset are reproduced. The best results are in bold; the second-best results are underlined.

Method	CUB-200-2011		meta-iNat		tiered-meta-iNat		Aircraft	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNet¤ (NeurIPS 2017)	82.98	91.38	76.47	90.20	47.61	71.06	66.47	82.15
DN4† (CVPR 2019)	85.25	92.84	80.64	92.00	51.34	73.56	86.44	91.92
DSN¤ (CVPR 2020)	80.80	91.19	78.80	89.77	46.61	72.79	58.03	69.85
CTX¤ (NeurIPS 2020)	78.47	90.90	69.04	88.39	42.91	69.88	79.24	85.97
DeepEMD¤ (CVPR 2020)	83.35	91.60	76.05	86.82	48.14	66.27	61.86	76.17
FRN¤ (CVPR 2021)	83.16	92.59	73.52	91.83	48.86	76.62	70.63	83.84
IEPT† (ICLR 2021)	77.60	87.64	67.34	82.69	49.67	68.60	71.08	83.41
DMN4† (AAAI 2022)	85.97	92.54	80.31	91.77	48.38	70.88	86.73	93.61
FRN+TDM† (CVPR 2022)	82.41	92.37	81.43	92.66	49.85	74.87	70.35	83.36
MCL-Katz¤ (CVPR 2022)	85.63	93.18	79.34	91.84	49.68	76.05	87.69	93.28
LCCRN† (TCSVT 2023)	82.99	93.52	77.56	91.14	48.72	74.61	88.48	94.61
ESPT† (AAAI 2023)	85.22	94.12	76.86	90.78	45.66	72.97	85.22	94.12
Bi-FRN† (AAAI 2023)	82.90	93.11	76.80	90.38	51.66	72.88	87.05	93.78
C2-Net† (AAAI 2024)	83.37	92.20	70.98	91.14	45.85	72.25	75.50	87.65
SRM† (PR 2024)	86.42	93.70	80.24	92.05	47.79	73.62	87.70	93.78
CDN4 (**Ours**)	**86.47**	**94.58**	**82.05**	**93.07**	**54.76**	**78.09**	**88.98**	**95.32**

learning rate of 0.1. We train our model from scratch for 600 episodic epochs and scale down the learning rate by 10 at epochs 300, 400, and 500. Each episode comprises an N-way K-shot task. Our experiments adopt 10-way 5-shot or 10-way 1-shot training setups, corresponding to 5-way 5-shot or 5-way 1-shot testing setups. For the experiments on meta-iNat and tiered-meta-iNat, we train the model for 100 epochs using the Adam optimizer. The initial learning rate is set to 0.001, and a learning rate decay of 0.5 is applied every 20 epochs. The method is implemented in PyTorch using one NVIDIA GeForce RTX 4090 with 24 GB memory.

Evaluation. During the test stage, we evaluate the performance of our approach on the standard 5-way 1-shot and 5-way 5-shot scenarios. We randomly sample 10,000 episodes from the test set to conduct each experiment and use the average accuracy as the evaluation metric. In each episode, we randomly select 15 query images per class for evaluation in 5-way 1-shot and 5-way 5-shot tasks.

4.3. Comparison with state-of-the-arts methods

The effectiveness of our proposed CDN4 is evaluated by comparing it with several representative and state-of-the-art methods, all with the ResNet-12 backbone as the feature extractor. The comparison methods include two global feature-based methods, namely ProtoNet [5] and DSN [7], and 12 spatially local feature-based methods, namely DN4 [9] and its improved method (DMN4 [21]), FRN [10] and its improved methods (TDM [22], LCCRN [36], Bi-FRN [20]), CTX [37], DeepEMD [8], MCL-Katz [11], IEPT [16], ESPT [17], C2-Net [38], and SRM [14].

The results of the experiments are listed in Table 2. We can see that our approach consistently performs better than state-of-the-art approaches on all three datasets. Specifically, it can be observed that our proposed CDN4 achieves the state-of-the-art results: 86.47% (1-shot) and 94.58% (5-shot) on CUB-200-2011, 82.05% (1-shot) and 93.07% (5-shot) on meta-iNat, 54.76% (1-shot) and 78.09% (5-shot) on tiered-meta-iNat, and 88.98% (1-shot) and 94.43% (5-shot) on Aircraft. Furthermore, the advantage of our CDN4 is more pronounced on the more difficult dataset tiered-meta-iNat as its accuracy is higher than the second best by 3.10% in the 1-shot setting and 3.22% in the 5-shot setting.

In sum, the experimental results show that our CDN4 has achieved optimal performance in fine-grained few-shot image classification tasks. This indicates that the proposed model can capture stronger transferable and discriminative features and have better generalization, by introducing more intra-class variations through the designed modules.

Table 3
Ablation studies on the effectiveness of the Episodic Dual-Branch Structure (EDBS) and Cross-view Local Metric Module (CLMM). Five-way classification accuracy on CUB-200-2011, meta-iNat and tiered-meta-iNat datasets were reported.

Method	CUB-200-2011		meta-iNat		tiered-meta-iNat	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Baseline (DN4)	85.25	92.84	80.64	92.00	51.34	73.56
+ EDBS	86.14	94.46	81.08	92.92	54.12	77.71
+ EDBS + CLMM (CDN4)	**86.47**	**94.58**	**82.05**	**93.07**	**54.76**	**78.09**

4.4. Ablation studies

4.4.1. Effectiveness of individual modules

To evaluate the effectiveness of the Episodic Dual-Branch Structure (EDBS) and the Cross-view Local Metrics Module (CLMM), we conducted the following two experiments:

1. On top of DN4 (baseline), add EDBS: This module constructs two branches by introducing random geometric transformations. After that, we compute the normalized similarity by using only the original support–original query and transformed support–transformed query data pairs, i.e., including only s_1 and s_4 in Eq. (9). The loss function (Eq. (11)) is set as the average of $\mathcal{L}_1$ and $\mathcal{L}_4$.
2. On top of DN4 + EDBS, add CLMM: Since this module computes the similarity between local support and query features across the two branches, it cannot be implemented alone. Therefore, we add it on top of DN4 + EDBS, which is same as the proposed CDN4. It verifies the effectiveness of normalizing the similarity across the four cross-view data pairs and including $\mathcal{L}_2$ and $\mathcal{L}_3$ into the loss function.

As shown in Table 3, the classification accuracy is greatly improved after the introduction of EDBS. This highlights the importance of increasing intra-class variation to enhance classification performance. The accuracy also improves slightly after the introduction of the two cross-view measurement pairs of CLMM. This indicates that, by requiring the model to achieve high similarity under four views, particularly those with larger intra-class differences created by pairing the original image with its transformed version, it encourages the learning of more discriminative features that remain consistent across different views.

4.4.2. Influence of random geometric transformations

The EDBS applies a random geometric transformation $\mathcal{T}$ to obtain the transformed episode. In this section, we discuss the influence of different transformation methods on classification accuracy.

As shown in Table 4, the first four rows show the performance when we switch the rotation operation to the other single transformations:

Table 4
The five-way classification accuracy comparison using different random transformation functions on CUB-200-2011, meta-iNat, and tiered-meta-iNat dataset. The first four rows show when a single transformation function is adopted, and the next two show when a hybrid transformation function is adopted.

Transformation	CUB-200-2011		meta-iNat		tiered-meta-iNat	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Random Erasing	86.17	94.27	80.76	92.33	49.65	73.70
Mixup	84.59	93.52	80.91	92.04	50.13	74.92
Shear	85.54	93.44	81.62	92.63	50.82	74.92
Flip	87.55	94.30	82.44	93.02	53.17	76.78
Rotation & Flip	84.25	94.31	81.48	93.06	53.85	75.67
Rotation | Flip	**86.57**	**94.71**	**82.71**	**93.24**	54.23	77.97
Rotation (Ours)	86.47	94.58	82.05	93.07	**54.76**	**78.09**

Table 5
The five-way classification accuracy comparison using different rotation sets on the CUB-200-2011, meta-iNat, and tiered-meta-iNat dataset.

Rotation set			CUB-200-2011		meta-iNat		tiered-meta-iNat	
90°	180°	270°	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
✓	–	–	86.65	93.56	**83.53**	**93.50**	53.99	77.42
–	✓	–	**86.75**	94.53	82.18	92.98	53.46	76.96
–	–	✓	83.78	94.78	83.25	93.15	53.98	77.42
✓	✓	–	86.71	94.19	82.15	93.10	54.73	**78.64**
✓	–	✓	86.68	94.79	82.60	93.32	**54.87**	78.16
–	✓	✓	85.24	**94.81**	82.50	93.08	54.78	78.14
✓	✓	✓	86.47	94.58	82.05	93.07	54.76	78.09

random erasing, mixup, shear, and flip. It can be observed that using random erasing, mixup or shear is inferior to using rotation as adopted in this paper, while using flip achieves similar results. This is likely due to the fact that discriminative features often occupy only a small region in the image and are thus susceptible to distortions with significant alternations. Then, we try to combine flip and rotation since they both outperform other transformations. In the fifth row of Table 4, we apply both rotation and flip to the original episode, while in the sixth row, we randomly select either rotation or flip. By comparing these two rows, it is evident that using only one transformation at a time is superior to using both transformations simultaneously. Finally, compared with our method which uses rotation alone, randomly applying flip or rotation shows better performance on the CUB-200-2011 and meta-iNat datasets but worse performance on the more complex dataset tiered-meta-iNat. The latter is likely because more variable transformations may mislead the model and make it difficult to learn more discriminative features. The difference between these two choices is relatively small, and for simplicity, we opt to use rotation only. In sum, this ablation study shows that introducing larger intra-class variation in the transformed branches enables the model to learn more discriminative local features.

4.4.3. Influence of the rotation set

In CDN4, we use a random geometric transformation $T \in \{t_r | r = 1, \ldots, 3\}$ to create a transformed branch from the original branch, which randomly rotates all images in the episode by $r \times 90°$. This subsection will investigate how different rotation sets affect classification accuracy. As shown in Table 5, we examine the use of a single rotation transformation in the first three rows, i.e., rotating the image by 90°, 180°, or 270°, randomly choosing from two rotation angles in rows 4–6, and randomly choosing from three rotation angles (i.e., our method) in the last row. It can be seen that no particular rotation set outperforms others in terms of classification accuracy, and the accuracy across these cases is quite similar. This suggests that the proposed approach is stable under varying choices of rotation angles.

4.4.4. Influence of cross-view measurements

In CLMM, four cross-view measurements are constructed, from which four cross-entropy losses are computed and then combined to generate the cross-view loss $\mathcal{L}_{cross}$. In this subsection, we will evaluate the influence of different data pairs on the classification accuracy.

Table 6
The five-way classification accuracy comparison using different loss functions in CLMM on the CUB-200-2011, meta-iNat, and tiered-meta-iNat dataset.

$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_4$	CUB-200-2011		meta-iNat		tiered-meta-iNat	
				1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
✓	–	–	–	85.51	93.54	80.82	92.04	51.34	73.56
–	✓	–	–	84.95	92.20	76.72	90.96	50.44	73.62
–	–	✓	–	85.02	91.00	76.72	90.96	50.69	73.12
–	–	–	✓	84.51	92.64	78.73	91.61	52.68	77.28
✓	–	–	✓	86.14	94.46	81.58	92.92	54.12	77.71
–	✓	✓	–	84.25	93.41	80.33	92.43	53.46	76.91
✓	✓	✓	✓	**86.47**	**94.58**	**82.05**	**93.07**	**54.76**	**78.09**

Table 7
The five-way classification accuracy comparison using the different similarity measures on the CUB-200-2011, meta-iNat, and tiered-meta-iNat dataset. For Euclidean, Manhattan, and Chebyshev distances, a learnable scaling factor is included; for cosine similarity, performance with and without a scaling factor is reported.

Similarity measure	CUB-200-2011		meta-iNat		tiered-meta-iNat	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Euclidean	81.43	92.04	79.01	91.45	49.88	75.34
Manhattan	73.88	85.64	75.89	90.30	44.12	73.56
Chebyshev	71.99	83.13	67.20	83.20	38.91	61.05
Cosine (w/ scaling)	**86.80**	**95.02**	81.81	**93.12**	54.12	**78.43**
Cosine (w/o scaling)	86.47	94.58	**82.05**	93.07	**54.76**	78.09

As shown in Table 6, the first four rows report scenarios in which a single loss function is used to train the network: $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, or $\mathcal{L}_4$; the fifth row uses only $\mathcal{L}_1$ and $\mathcal{L}_4$, where the support and query samples used to compute these two losses come from the same branch; the sixth row uses only $\mathcal{L}_2$ and $\mathcal{L}_3$, where the support and query samples corresponding to these two losses come from different branches; and the last row is the method proposed in this paper, which averages $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, and $\mathcal{L}_4$ to generate $\mathcal{L}_{cross}$. By comparing these results, we can make the following observations.

First, the combined loss function $\mathcal{L}_1 + \mathcal{L}_4$ outperforms either loss function alone, and $\mathcal{L}_2 + \mathcal{L}_3$ is superior to the individual losses in five out of six cases. This indicates that increasing the number of similarity comparisons can help learn more discriminative features. The particularly low performance (84.25%) in the one-shot case on CUB-200-2011 may be due to the small sample size, which likely limits the ability to learn discriminative features that can overcome the increased intra-class variation we introduced. Secondly, we find that performance using $\mathcal{L}_1$ alone is better than using $\mathcal{L}_2$, $\mathcal{L}_3$, or $\mathcal{L}_4$ alone, and that the combination of $\mathcal{L}_1 + \mathcal{L}_4$ outperforms $\mathcal{L}_2 + \mathcal{L}_3$. This suggests the importance of retaining the original support-original query pair, which is generally easier, before introducing more diverse data. In other words, starting with harder tasks might hinder the learning process. Thirdly, compared with two losses, using all four losses can further improve the performance. The cross-view measurement increases the intra-class variation through the interaction between the original and transformation branches, leading the model to pay more attention to the discriminative features.

4.4.5. Influence of the similarity measure

Another important factor in CLMM is the similarity measure between local features of support and query samples, which we currently employ the cosine similarity. In this subsection, we investigate this by evaluating three other distance metrics, namely the Euclidean distance, the Manhattan distance, and the Chebyshev distance. Moreover, according to [39], networks optimized directly with the cosine similarity may fail to converge due to its limited value range, and it is recommended to incorporate a scaling parameter to address this issue. Therefore, for each similarity measure, we consider their performance with and

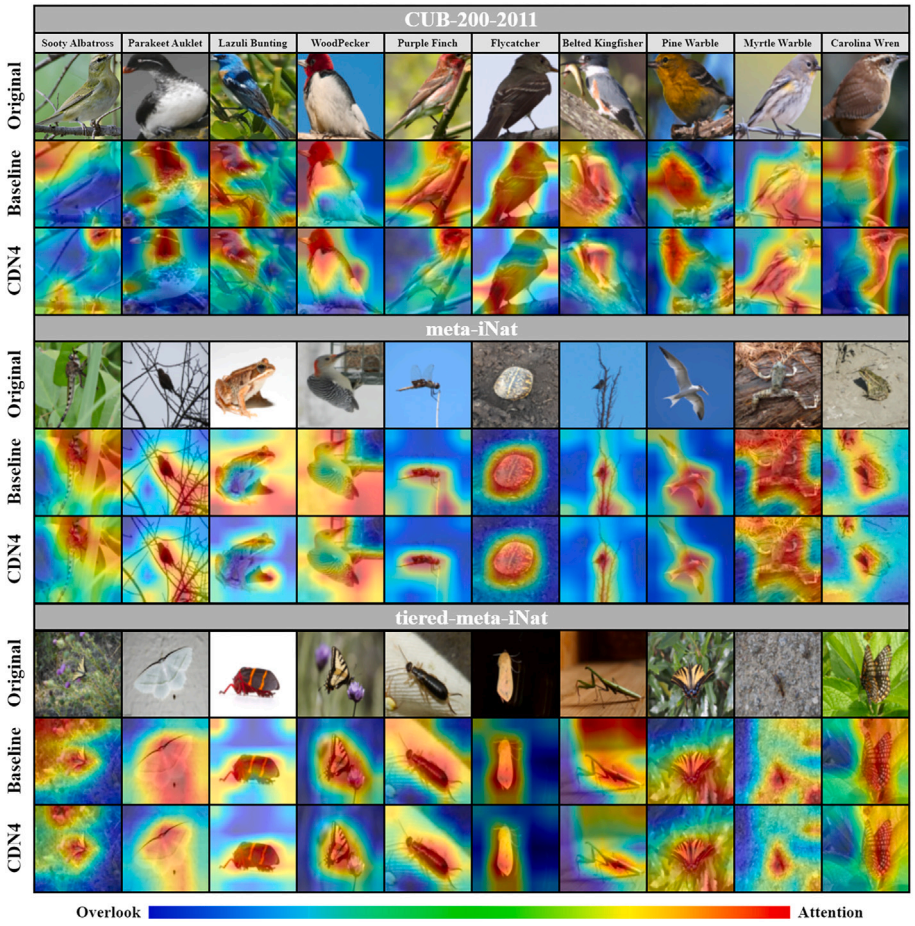

Fig. 3. The Grad-CAM class activation map of query samples on the CUB-200-2011 dataset, meta-iNat dataset, and tiered-meta-iNat dataset. The first row corresponds to the original images, and the second and third rows correspond to the features from the baseline model DN4 and our proposed CDN4. Compared with DN4, CDN4 pays more accurate and focused attention to the target-related discriminative regions.

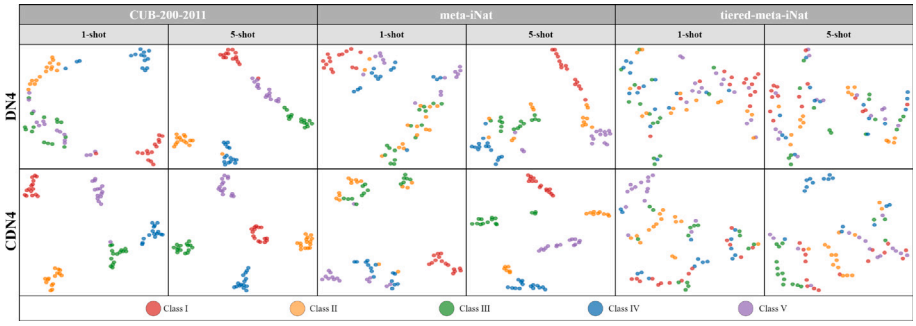

Fig. 4. Visualization of feature embeddings of query samples obtained by DN4 and CDN4 on the CUB-200-2011, meta-iNat, and tiered-meta-iNat datasets, where different colors represent different categories.

without a learnable scaling factor. The factor is a scalar introduced in Eq. (9), which can optimized simultaneously with other network parameters during training. Table 7 lists the results; due to the poor performance of Euclidean, Manhattan, and Chebyshev distances without scaling, their accuracies are not reported. Firstly, we see that a learnable scaling factor is valuable, as it improves the classification accuracy of cosine similarity in four out of six cases. Secondly, cosine similarity, with or without scaling, consistently outperforms the other metrics. This seems to align with previous research, which suggests that cosine similarity has intrinsic consistency with the softmax loss [40] and encourages it to focus on maximizing the angular separability [39].

4.5. Visualization

4.5.1. Grad-CAM

To gain deeper insight into how CDN4 works, we utilize Grad-CAM [41] to visualize the areas within the query samples that are important for classification. The heat maps are displayed in Fig. 3. In each block, the first row shows the original image, followed by the feature maps of the baseline DN4 and our method in the second and third rows, respectively. The red region in each image corresponds to the region where the model pays more attention to, and the warmer the color, the higher the attention.

Table 8
Comparison of the number of parameters, the number of floating-point operations, training time, inference time, and memory on the CUB-200-2011 dataset.

Methods	Params (M)		FLOPs (G)		Train time (s)		Inference time (s)		Memory (MB)	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNet	12.42	12.42	845.52	1056.90	0.26	0.26	0.035	0.037	14 454	20 844
DN4	12.42	12.42	563.68	704.60	0.22	0.26	0.034	0.036	17 810	21 520
FRN	12.42	12.42	845.52	1056.90	0.26	0.29	0.048	0.051	14 616	16 480
DeepEMD	12.42	12.42	563.68	704.60	0.98	0.99	0.167	1.657	8936	10 770
DMN4	12.43	12.43	563.68	704.60	0.25	0.28	0.035	0.037	8963	10 770
MCL	12.42	12.42	563.68	704.60	0.20	0.28	0.036	0.041	9170	11 004
ESPT	12.42	12.42	563.68	704.60	0.20	0.32	0.031	0.038	8662	10 032
C2-Net	18.49	18.49	577.35	720.40	0.17	0.23	0.030	0.033	12 186	13 008
BiFRN	16.12	16.12	867.69	1084.46	0.27	0.38	0.026	0.033	15 714	23 756
CDN4	12.42	12.42	1127.36	1409.20	0.44	0.52	0.061	0.070	20 084	23 794

From this figure, we see that DN4 can identify the target objects in most cases, but it often assigns high importance to the entire target and sometimes includes the background as well. In contrast, our CDN4 model can identify the target in all cases, and more importantly, it only focuses on small, important regions, such as the eye, beak, and tail of the bird. Such a stark contrast in the attention demonstrates the effectiveness of cross-view measurement in extracting discriminative local features, which play a key role in fine-grained few-shot image classification.

4.5.2. t-SNE

We also use t-SNE [42] to visualize the distribution of query samples in the embedded space. As shown in Fig. 4, compared with DN4, intra-class samples of CDN4 are more clustered, while inter-class samples are more distant. It shows that after introducing more intra-class variation, the feature representation learned by the model is more conducive to the improvement of generalization performance.

4.6. Comparison of computational cost

To understand the computational cost of CDN4, we measure the number of parameters, the number of floating-point operations (FLOPs), training time, inference time, and memory usage. The FLOPs are measured during the training phase in the 10-way 1-shot and 10-way 5-shot tasks with 15 query samples per class using the CUB-200-2011 dataset. Training time and inference time are calculated as the average time required for 100 randomly selected episodes and are reported in seconds [11]. Memory usage refers to the GPU memory consumed by the model during training and is reported in MB.

In Table 8, the number of parameters and FLOPs for some methods appear the same since we used the same code framework when running the experiment. Compared with the baseline DN4, CDN4 does not introduce more parameters. Its FLOPs, training time, and inference time are twice those of DN4, and memory usage is 12.8% higher in the one-shot case and 10.6% higher in the five-shot case. The increase in computational cost comes from two sources: the EDBS generates transformed episodes on top of the original episodes, which doubles the amount of data, and the CLMM module measures the similarity between four pairs instead of one pair in DN4.

5. Conclusion

In this study, we propose a novel Cross-view Deep Nearest Neighbor Neural Network (CDN4) for fine-grained few-shot image classification. Our approach integrates the Episodic Dual-Branch Structure and the Cross-view Local Metric Module, which enables us to obtain more discriminative features and achieve a more accurate metric by building cross-view measurements between dual branches. Extensive experiments, including ablation studies and visualization, suggest that our method achieves superior classification accuracy and generalization, as well as stable performance under different augmentations. In the future, we plan to investigate the applicability of our method to pre-trained foundation models such as CLIP. Additionally, noting the longer training and inference time, we plan to parallelize the feature extraction and similarity calculation processes.

CRediT authorship contribution statement

Xiaoxu Li: Supervision, Resources, Methodology. **Shuo Ding:** Writing – original draft, Validation, Software. **Jiyang Xie:** Investigation, Data curation. **Xiaochen Yang:** Writing – review & editing, Visualization, Formal analysis. **Zhanyu Ma:** Writing – review & editing, Resources, Conceptualization. **Jing-Hao Xue:** Writing – review & editing, Supervision, Project administration.

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.

Acknowledgments

This work was supported in part by the Beijing Natural Science Foundation Project No. L242025, in part by the Royal Society under International Exchanges Award IEC\NSFC\201071, IEC\NSFC\211131, in part by the National Natural Science Foundation of China (NSFC) under Grant 62176110, 62225601, U23B2052, in part by the Key Talent Program of Gansu Province under Grant 2025RCXM002, the S&T Program of Hebei under Grant SZX2020034, and Hong-liu Distinguished Young Talents Foundation of Lanzhou University of Technology, China.

Data availability

All datasets used are publicly available. Code for CDN4 is available at https://github.com/ShawnTing1/CDN4.

References

[1] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, Adv. Neural Inf. Process. Syst. 29 (2016).
[2] X. Li, Z. Sun, J.-H. Xue, Z. Ma, A concise review of recent few-shot meta-learning methods, Neurocomputing 456 (2021) 463–468.
[3] X. Li, X. Yang, Z. Ma, J.-H. Xue, Deep metric learning for few-shot image classification: A review of recent developments, Pattern Recognit. 138 (2023) 109381.
[4] Z. Li, Z. Hu, W. Luo, X. Hu, SaberNet: Self-attention based effective relation network for few-shot learning, Pattern Recognit. 133 (2023) 109024.
[5] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, Adv. Neural Inf. Process. Syst. 30 (2017).
[6] F. Sung, Y. Yang, L. Zhang, T. Xiang, P.H. Torr, 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.

[7] C. Simon, P. Koniusz, R. Nock, M. Harandi, Adaptive subspaces for few-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4136–4145.
[8] C. Zhang, Y. Cai, G. Lin, C. Shen, Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12203–12213.
[9] 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 7260–7268.
[10] D. Wertheimer, L. Tang, B. Hariharan, Few-shot classification with feature map reconstruction networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8012–8021.
[11] Y. Liu, W. Zhang, C. Xiang, T. Zheng, D. Cai, X. He, Learning to affiliate: Mutual centralized learning for few-shot classification, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 14411–14420.
[12] H. Tang, C. Yuan, Z. Li, J. Tang, Learning attention-guided pyramidal features for few-shot fine-grained recognition, Pattern Recognit. 130 (2022) 108792.
[13] Z. Li, Y. Sun, L. Zhang, J. Tang, CTNet: Context-based tandem network for semantic segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 44 (12) (2021) 9904–9917.
[14] X. Li, Z. Li, J. Xie, X. Yang, J.-H. Xue, Z. Ma, Self-reconstruction network for fine-grained few-shot classification, Pattern Recognit. 153 (2024) 110485.
[15] O. Boiman, E. Shechtman, M. Irani, In defense of nearest-neighbor based image classification, in: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2008, pp. 1–8.
[16] M. Zhang, J. Zhang, Z. Lu, T. Xiang, M. Ding, S. Huang, IEPT: Instance-level and episode-level pretext tasks for few-shot learning, in: International Conference on Learning Representations, 2021.
[17] Y. Rong, X. Lu, Z. Sun, Y. Chen, S. Xiong, ESPT: a self-supervised episodic spatial pretext task for improving few-shot learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, No. 8, 2023, pp. 9596–9605.
[18] 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.
[19] A. Raghu, M. Raghu, S. Bengio, O. Vinyals, Rapid learning or feature reuse? Towards understanding the effectiveness of MAML, in: International Conference on Learning Representations, 2020.
[20] J. Wu, D. Chang, A. Sain, X. Li, Z. Ma, J. Cao, J. Guo, Y.-Z. Song, Bi-directional feature reconstruction network for fine-grained few-shot image classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, No. 3, 2023, pp. 2821–2829.
[21] Y. Liu, T. Zheng, J. Song, D. Cai, X. He, DMN4: Few-shot learning via discriminative mutual nearest neighbor neural network, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, No. 2, 2022, pp. 1828–1836.
[22] S. Lee, W. Moon, J.-P. Heo, Task discrepancy maximization for fine-grained few-shot classification, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2022, pp. 5331–5340.
[23] T.D. Kulkarni, W.F. Whitney, P. Kohli, J.B. Tenenbaum, Deep convolutional inverse graphics network, Adv. Neural Inf. Process. Syst. (2015) 2539–2547.
[24] H. Zhang, M. Cissé, Y.N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, in: International Conference on Learning Representations, 2018.
[25] Z. Yang, J. Wang, Y. Zhu, Few-shot classification with contrastive learning, in: European Conference on Computer Vision, Springer, 2022, pp. 293–309.
[26] F. Liu, S. Yang, D. Chen, H. Huang, J. Zhou, Few-shot classification guided by generalization error bound, Pattern Recognit. 145 (2024) 109904.
[27] Z. Fang, X. Jiang, H. Tang, Z. Li, Learning contrastive self-distillation for ultra-fine-grained visual categorization targeting limited samples, IEEE Trans. Circuits Syst. Video Technol. (2024).
[28] Z. Peng, Z. Li, J. Zhang, Y. Li, G.-J. Qi, J. Tang, Few-shot image recognition with knowledge transfer, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 441–449.
[29] H.-J. Ye, H. Hu, D.-C. Zhan, F. Sha, Few-shot learning via embedding adaptation with set-to-set functions, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8808–8817.
[30] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200–2011 Dataset, California Institute of Technology, 2011.
[31] D. Wertheimer, B. Hariharan, Few-shot learning with localization in realistic settings, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 6558–6567.
[32] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, A. Vedaldi, Fine-grained visual classification of aircraft, 2013, arXiv preprint arXiv:1306.5151.
[33] 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, 2019.
[34] Y. Zhu, C. Liu, S. Jiang, et al., Multi-attention meta learning for few-shot fine-grained image recognition, in: International Joint Conference on Artificial Intelligence, 2020, pp. 1090–1096.
[35] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[36] 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. 33 (2023) 7530–7540.
[37] C. Doersch, A. Gupta, A. Zisserman, Crosstransformers: spatially-aware few-shot transfer, Adv. Neural Inf. Process. Syst. 33 (2020) 21981–21993.
[38] Z.-X. Ma, Z.-D. Chen, L.-J. Zhao, Z.-C. Zhang, X. Luo, X.-S. Xu, Cross-layer and cross-sample feature optimization network for few-shot fine-grained image classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, No. 5, 2024, pp. 4136–4144.
[39] F. Wang, X. Xiang, J. Cheng, A.L. Yuille, Normface: L2 hypersphere embedding for face verification, in: Proceedings of the 25th ACM International Conference on Multimedia, 2017, pp. 1041–1049.
[40] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, W. Liu, CosFace: Large margin cosine loss for deep face recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5265–5274.
[41] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 618–626.
[42] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (11) (2008).

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.

Shuo Ding received his B.Eng degree in Civil Engineering from Lanzhou University of Technology in 2022. He is currently pursuing a master's degree in Computer Application Technology at Lanzhou University of Technology. His research interests include machine learning, computer vision, and few-shot learning.

Jiyang Xie received the B.E. degree in information engineering and the Ph.D. degree in information and communication engineering from the Beijing University of Posts and Telecommunications (BUPT), China, in 2017 and 2022, respectively. 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, London, in 2020. She is currently a Lecturer with the School of Mathematics and Statistics, University of Glasgow, Glasgow, U.K. Her research interests include statistical classification, metric learning, and hyperspectral image analysis. She is an Associate Editor of Neurocomputing and IEEE Transactions on Circuits and Systems for Video Technology.

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.

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 of Statistical Pattern Recognition in the Department of Statistical Science, University College London. His research interests include statistical pattern recognition, machine learning, and computer vision. He received the Best Associate Editor Award of 2021 from the IEEE Transactions on Circuits and Systems for Video Technology, and the Outstanding Associate Editor Award of 2022 from the IEEE Transactions on Neural Networks and Learning Systems.