Joint Distribution Matters: Deep Brownian Distance Covariance for Few-Shot Classification

Jiangtao Xie[1,*], Fei Long[1,*], Jiaming Lv[1], Qilong Wang[2], Peihua Li[1,†]
[1]Dalian University of Technology, China [2]Tianjin University, China

Abstract

Few-shot classification is a challenging problem as only very few training examples are given for each new task. One of the effective research lines to address this challenge focuses on learning deep representations driven by a similarity measure between a query image and few support images of some class. Statistically, this amounts to measure the dependency of image features, viewed as random vectors in a high-dimensional embedding space. Previous methods either only use marginal distributions without considering joint distributions, suffering from limited representation capability, or are computationally expensive though harnessing joint distributions. In this paper, we propose a deep Brownian Distance Covariance (DeepBDC) method for few-shot classification. The central idea of DeepBDC is to learn image representations by measuring the discrepancy between joint characteristic functions of embedded features and product of the marginals. As the BDC metric is decoupled, we formulate it as a highly modular and efficient layer. Furthermore, we instantiate DeepBDC in two different few-shot classification frameworks. We make experiments on six standard few-shot image benchmarks, covering general object recognition, fine-grained categorization and cross-domain classification. Extensive evaluations show our DeepBDC significantly outperforms the counterparts, while establishing new state-of-the-art results. The source code is available at http://www.peihuali.org/DeepBDC.

1. Introduction

Few-shot classification [15, 17] is concerned with a task where a classifier can be adapted to distinguish classes unseen previously, given only a very limited number of examples of these classes. This is a challenging problem as scarcely labeled examples are far from sufficient for learning abundant knowledge and also likely lead to overfitting. One practical solution is based on the technique of meta-learning or learning to learn [12, 39], in which the episodic training is formulated to transfer the knowledge obtained on a massive meta-training set spanning a large number of known classes to the few-shot regime of novel classes. Among great advances that have been made, the line of metric-based methods attracts considerable research interest [15, 26, 33, 39], achieving state-of-the-art performance [45, 47] in recent years.

The primary idea of the metric-based few-shot classification is to learn representations through deep networks, driven by the similarity measures between a query image and few support images of some class [33, 47]. Statistically, the features of a query image (resp., support images) can be viewed as observations of a random vector X (resp., Y) in a high-dimensional embedding space. Therefore, the similarity between images can be measured by means of probability distributions. However, modeling distributions of high-dimensional (and often few) features is hard and a common method is to model statistical moments. ProtoNet [33] and its variants (e.g., [26]) represent images by first moment (mean vector) and use Euclidean distance or cosine similarity for metric learning. To capture richer statistics, several works study second moment (covariance matrix) [44] or combination of first and second moments in the form of Gaussians [20] for image representations, while adopting Frobenius norm or Kullback-Leiberler (KL) divergence as similarity measures. However, these methods only exploit marginal distributions while neglecting joint distributions, limiting the performance of learned models. In addition, the covariances can only model linear relations.

In general, the dependency between X and Y should be measured in light of their joint distribution $f_{XY}(\mathbf{x}, \mathbf{y})$ [6]. Earth Mover's Distance (EMD) is an effective method for measuring such dependency. As described in [29, Sec. 2.3], EMD seeks an optimal joint distribution $f_{XY}(\mathbf{x}, \mathbf{y})$, whose marginals are constrained to be given $f_X(\mathbf{x})$ and $f_Y(\mathbf{y})$, so that the expectation of transportation cost is minimal. In few-shot classification, DeepEMD [47] proposes differential EMD for optimal matching of image regions. Though achieving state-of-the-art performance, DeepEMD is computationally expensive [45], due to inherent linear program-

[*] Equal contribution. [†] Corresponding author, peihuali@dlut.edu.cn. The work was supported by National Natural Science Foundation of China (61971086, 61806140), and CCF-Baidu Open Fund (2021PP15002000).

Method	Probability model	Dis-similarity/similarity measure	Joint distribution	Dependency	Latency	Accuracy (%)	
						1-shot	5-shot
ProtoNet [33]	Mean vector	$\|\boldsymbol{\mu}_X - \boldsymbol{\mu}_Y\|^2$ or $\frac{\boldsymbol{\mu}_X^T \boldsymbol{\mu}_Y}{\|\boldsymbol{\mu}_X\|\|\boldsymbol{\mu}_Y\|}$	No	N/A	Low	49.42	68.20
CovNet [44]	Covariance matrix	$\|\boldsymbol{\Sigma}_X - \boldsymbol{\Sigma}_Y\|^2$	No	Linear	Low	49.64	69.45
ADM [20]	Gaussian distribution	$D_{\mathrm{KL}}(\mathcal{N}_{\boldsymbol{\mu}_X,\boldsymbol{\Sigma}_X} \| \mathcal{N}_{\boldsymbol{\mu}_Y,\boldsymbol{\Sigma}_Y})$	No	N/A	Low	53.10	69.73
DeepEMD [47]	Discrete distribution	$\min_{f_{\mathbf{x}_j,\mathbf{y}_l} \geq 0} \sum_j \sum_l f_{\mathbf{x}_j,\mathbf{y}_l} c_{\mathbf{x}_j,\mathbf{y}_l}$ s.t. $\sum_l f_{\mathbf{x}_j,\mathbf{y}_l} = f_{\mathbf{x}_j}, \sum_j f_{\mathbf{x}_j,\mathbf{y}_l} = f_{\mathbf{y}_l}$ for $\forall j, l$	Yes	N/A	High	65.91	82.41
DeepBDC (ours)	Characteristic function	$\int_{\mathbb{R}^p} \int_{\mathbb{R}^q} \frac{\|\phi_{XY}(\mathbf{t},\mathbf{s}) - \phi_X(\mathbf{t})\phi_Y(\mathbf{s})\|^2}{c_p c_q \|\mathbf{t}\|^{1+p} \|\mathbf{s}\|^{1+q}} d\mathbf{t} d\mathbf{s}$	Yes	Nonlinear & Independence	Low	**67.34**	**84.46**

Table 1. Comparison between our DeepBDC and the counterparts. To quantify the dependency between random vectors X and Y, moments based methods [20, 33, 44] only model marginal distributions, suffering from limited representation capability; though achieving state-of-the-art performance by considering joint distributions, DeepEMD [47] is computationally expensive. Our DeepBDC measures discrepancy between joint characteristic function and product of the marginals, which can be efficiently computed in closed-form, and model non-linear relations and fully characterizes independence. Note that for a random vector its characteristic function and probability distribution are equivalent in that they form a Fourier transform pair. Here we report accuracies of 5-way 1-shot/5-shot classification on *mini*ImageNet; our result is obtained by Meta DeepBDC and results of the counterparts are duplicated from respective papers.

ming algorithm. Mutual information (MI) [3, 28] is a well-known measure, which can quantify the dependency of two random variables by KL-divergence between their joint distribution and product of the marginals. Unfortunately, computation of MI is difficult in real-valued, high-dimensional setting [2], and often involves difficult density modeling or lower-bound estimation of KL-divergence [14].

In this paper, we propose a deep Brownian Distance Covariance (DeepBDC) method for few-shot classification. The BDC metric, first proposed in [35, 36], is defined as the Euclidean distance between the joint characteristic function and product of the marginals. It can naturally quantify the dependency between two random variables. For discrete observations (features), the BDC metric is decoupled so that we can formulate BDC as a pooling layer, which can be seamlessly inserted into a deep network, accepting feature maps as input and outputting a BDC matrix as an image representation. In this way, the similarity between two images is computed as the inner product between the corresponding two BDC matrices. Therefore, the core of our DeepBDC is highly modular and plug-and-play for different methodologies of few-shot image classification. Specifically, we instantiate our DeepBDC in meta-learning framework (Meta DeepBDC), and in the simple transfer learning framework relying non-episodic training (STL DeepBDC). Contrary to covariance matrices, our DeepBDC can freely handle non-linear relations and fully characterize independence. Compared to EMD, it also considers joint distribution and above all, can be computed analytically and efficiently. Unlike MI, the BDC requires no density modeling. We present differences between our BDC and the counterparts in Tab. 1.

Our contributions are summarized as follows. (1) For the first time, we introduce Brownian distance covariance (BDC), a fundamental but largely overlooked dependency modeling method, into deep network-based few-shot classification. Our work suggests great potential and future applications of BDC in deep learning. (2) We formulate DeepBDC as a highly modular and efficient layer, suitable for different few-shot learning frameworks. Furthermore, we propose two instantiations for few-shot classification, i.e., Meta DeepBDC based on the meta-learning framework with ProtoNet as a blue print, and STL DeepBDC based on simple transfer learning framework without episodic training. (3) We perform thorough ablation study on our methods and conduct extensive experiments on six few-shot classification benchmarks. The experimental results demonstrate that both of our two instantiations achieve superior performance and meanwhile set new state-of-the-arts.

2. Related Works

Representation learning in few-shot classification The image representation and the similarity measure play important roles in few-shot classifications where only limited labeled examples are available. In light of the image representation, we can roughly divide the few-shot classification methods into two categories. In the first category, the image representations are based on distribution modeling. They use either first moment (mean vector) [33], second moment (covariance matrix) [44] , Gaussian distribution [20] or discrete probability [47], and, accordingly, adopts Euclidean distance (or cosine similarity), Frobenious norm, KL-divergence or Earther Mover's Distance as dis-similarity measures. The second category is concerned with feature reconstruction between the query image and the support images, by means of either directly linear reconstruction through Ridge regression [45] or attention mechanism [9, 46], or concerned with designing relational module to learn a transferable deep metric [34, 48]. Our methods belong to the first category, and the biggest difference from existing works is that we use Brownian Distance Covariance for representation learning in the few-shot regime.

Meta-learning versus simple transfer learning Meta-learning is a de facto framework for few-shot classification [12, 39]. It involves a family of tasks (episodes) split into disjoint meta-training and meta-testing sets. Typically, each task is formulated as a N-way K-shot classification, which spans N classes each provided with K support images and some query images. The meta-training and meta-testing sets share the episodic training strategy that facilitates generalization ability across tasks. Most of the methods, either optimization-based [12, 30] or metric-based [33, 34], follow this methodology. A lot of studies [5, 45, 47] have shown that, rather than meta-training from scratch, pre-training on the whole meta-training set is helpful for meta-learning. Recently, it has been found that simple transfer learning (STL) framework, which does not rely on episodic training at all, achieves very competitive performance [4, 8, 37]. For STL methods, during meta-training a deep network is trained for a common classification problem via standard cross-entropy loss on the whole meta-training set spanning all classes; during meta-testing, the trained model is used as an embedding model for feature extraction, then a linear model, such as a soft-max classifier [4, 8] or logistic regression model [37], is constructed and trained for the few-shot classification.

Finally, we mention that scarce works have ever used BDC in machine learning or computer vision, and so far we find one BDC-based dimension reduction method [7] which is not concerned with deep learning.

3. Proposed Method

In this section, we first introduce Brownian distance covariance (BDC). Then we formulate our DeepBDC in the convolutional networks. Finally, we instantiate our DeepBDC for few-shot image classification.

3.1. Brownian Distance Covariance (BDC)

The theory of BDC is first established in [35, 36] in light of characteristic function. The characteristic function of a random vector is equivalent to its probability density function (PDF), as they form a Fourier transform pair.

Let $X \in \mathbb{R}^p, Y \in \mathbb{R}^q$ be random vectors of dimension p and q, respectively, and let $f_{XY}(\mathbf{x},\mathbf{y})$ be their joint PDF. The joint characteristic function of X and Y is defined as

$$\phi_{XY}(\mathbf{t},\mathbf{s})=\int_{\mathbb{R}^p}\int_{\mathbb{R}^q}\exp(i(\mathbf{t}^T\mathbf{x}+\mathbf{s}^T\mathbf{y}))f_{XY}(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} \quad (1)$$

where i is the imaginary unit. Clearly, the marginal distributions of X and Y are respectively $\phi_X(\mathbf{t})=\phi_{XY}(\mathbf{t},\mathbf{0})$ and $\phi_Y(\mathbf{s})=\phi_{XY}(\mathbf{0},\mathbf{s})$ where $\mathbf{0}$ is a vector whose elements are all zero. From theory of probability, we know X and Y are independent if and only if $\phi_{XY}(\mathbf{t},\mathbf{s})=\phi_X(\mathbf{t})\phi_Y(\mathbf{s})$. Provided X and Y have finite first moments, the BDC metric is defined as

$$\rho(X,Y)=\int_{\mathbb{R}^p}\int_{\mathbb{R}^q}\frac{|\phi_{XY}(\mathbf{t},\mathbf{s})-\phi_X(\mathbf{t})\phi_Y(\mathbf{s})|^2}{c_p c_q\|\mathbf{t}\|^{1+p}\|\mathbf{s}\|^{1+q}}d\mathbf{t}d\mathbf{s} \quad (2)$$

where $\|\cdot\|$ denotes Euclidean norm, $c_p = \pi^{(1+p)/2}/\Gamma((1+p)/2)$ and Γ is the complete gamma function.

For the set of m observations $\{(\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_m,\mathbf{y}_m)\}$ which are independent and identically distributed (i.i.d.), a natural approach is to define the BDC metric in light of the empirical characteristic functions:

$$\phi_{XY}(\mathbf{t},\mathbf{s}) = \frac{1}{m}\sum_{k=1}^{m}\exp(i(\mathbf{t}^T\mathbf{x}_k + \mathbf{s}^T\mathbf{y}_k)) \quad (3)$$

Though Eq. (2) seems complicated, the BDC metric has a closed form expression for discrete observations. Let $\widehat{\mathbf{A}} = (\widehat{a}_{kl}) \in \mathbb{R}^{m\times m}$ where $\widehat{a}_{kl} = \|\mathbf{x}_k - \mathbf{x}_l\|$ be an Euclidean distance matrix computed between the pairs of observations of X. Similarly, we compute the Euclidean distance matrix $\widehat{\mathbf{B}} = (\widehat{b}_{kl}) \in \mathbb{R}^{m\times m}$ where $\widehat{b}_{kl} = \|\mathbf{y}_k - \mathbf{y}_l\|$. Then the BDC metric has the following form [35] [1]:

$$\rho(X,Y) = \mathrm{tr}\big(\mathbf{A}^T\mathbf{B}\big) \quad (4)$$

where $\mathrm{tr}(\cdot)$ denotes matrix trace, T denotes matrix transpose, and $\mathbf{A} = (a_{kl})$ is dubbed *BDC matrix*. Here $a_{kl} = \widehat{a}_{kl} - \frac{1}{m}\sum_{k=1}^{m}\widehat{a}_{kl} - \frac{1}{m}\sum_{l=1}^{m}\widehat{a}_{kl} - \frac{1}{m^2}\sum_{k=1}^{m}\sum_{l=1}^{m}\widehat{a}_{kl}$, where the last three terms indicate means of the l-th column, k-th row and all entries of $\widehat{\mathbf{A}}$, respectively. The matrix $\mathbf{B}$ can be computed in a similar manner from $\widehat{\mathbf{B}}$. As a BDC matrix is symmetric, $\rho(X,Y)$ can also be written as the inner product of two BDC vectors $\mathbf{a}$ and $\mathbf{b}$, i.e.,

$$\rho(X,Y) = \big\langle \mathbf{a},\mathbf{b}\big\rangle = \mathbf{a}^T\mathbf{b} \quad (5)$$

where $\mathbf{a}$ (resp., $\mathbf{b}$) is obtained by extracting the upper triangular portion of $\mathbf{A}$ (resp., $\mathbf{B}$) and then performing vectorization.

The metric $\rho(X,Y)$ has some desirable properties. (1) It is non-negative, and is equal to 0 if and only if X and Y are independent. (2) It can characterize linear and nonlinear dependency between X and Y. (3) It is invariant to individual translations and orthonormal transformations of X and Y, and equivariant to their individual scaling factors. That is, for any vectors $\mathbf{c}_1 \in \mathbb{R}^p, \mathbf{c}_2 \in \mathbb{R}^q$, scalars s_1, s_2 and orthonormal matrices $\mathbf{R}_1 \in \mathbb{R}^{p\times p}, \mathbf{R}_2 \in \mathbb{R}^{q\times q}$, $\rho(\mathbf{c}_1 + s_1\mathbf{R}_1X, \mathbf{c}_2 + s_2\mathbf{R}_2Y) = |s_1s_2|\rho(X,Y)$.

[1] Actually, $\rho(X,Y) = \frac{1}{m^2}\mathrm{tr}\big(\mathbf{A}^T\mathbf{B}\big)$ and the constant $\frac{1}{m^2}$ is assimilated into a learnable scaling parameter τ (see Sec. 3.3) and thus is left out.

3.2. Formulation of DeepBDC as a Pooling Layer

In terms of Eq. (4) and Eq. (5), we can see that the BDC metric is decoupled in the sense that we can independently compute the BDC matrix for every input image. Specifically, we design a two-layer module suitable for a convolutional network, which performs dimension reduction and computation of the BDC matrix, respectively. As the size of a BDC matrix increases quadratically with respect to the number of channels (feature maps) in the network, we insert a 1×1 convolution layer for dimension reduction right after the last convolution layer of the network backbone.

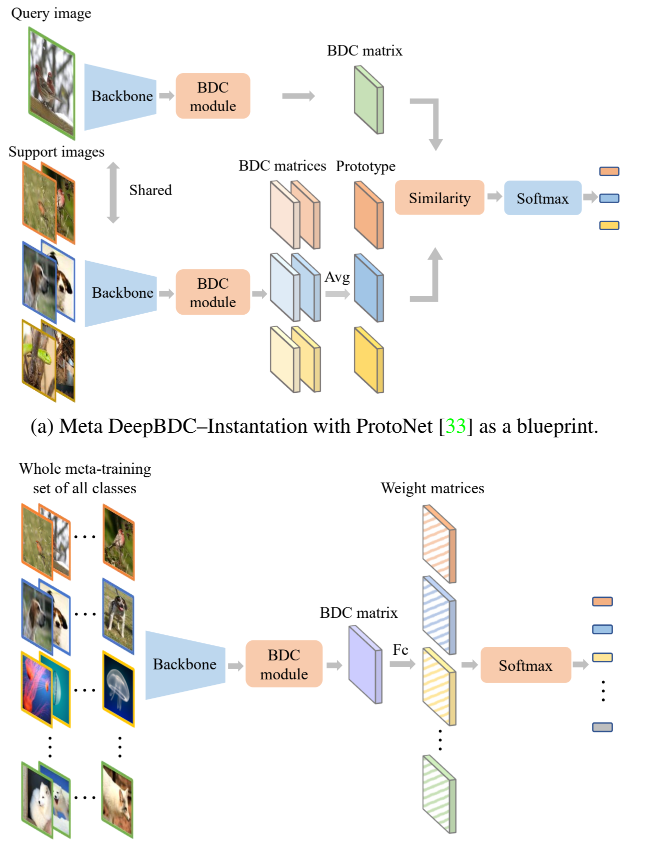
Suppose the network (including the dimension reduction layer) is parameterized by $\boldsymbol{\theta}$ which embeds a color image $\mathbf{z} \in \mathbb{R}^3$ into a feature space. The embedding of the image is a $h \times w \times d$ tensor, where h and w are spatial height and width while d is the number of channels. We reshape the tensor to a matrix $\mathbf{X} \in \mathbb{R}^{hw \times d}$, and can view either each column $\boldsymbol{\chi}_k \in \mathbb{R}^{hw}$ or each row (after transpose) $\mathbf{x}_j \in \mathbb{R}^d$ as an observation of random vector X. We mention that in practice, for either case the i.i.d. assumption may not hold, and comparison of the two options is given in Sec. 4.2.

In what follows, we take for example $\boldsymbol{\chi}_k$ as a random observation. We develop three operators, which successively compute the squared Euclidean distance matrix $\widetilde{\mathbf{A}} = (\tilde{a}_{kl})$ where $\tilde{a}_{kl}$ is squared Euclidean distance between the k-th column and l-th column of $\mathbf{X}$, the Euclidean distance matrix $\widehat{\mathbf{A}} = (\sqrt{\tilde{a}_{kl}})$, and the BDC matrix $\mathbf{A}$ obtained by subtracting from $\widehat{\mathbf{A}}$ its row mean, column mean and mean of all of its elements. That is,

$$\begin{aligned}
\widetilde{\mathbf{A}} &= 2\big(\mathbf{1}(\mathbf{X}^T\mathbf{X} \circ \mathbf{I})\big)_{\mathrm{sym}} - 2\mathbf{X}^T\mathbf{X} \\
\widehat{\mathbf{A}} &= \big(\sqrt{\tilde{a}_{kl}}\big) \\
\mathbf{A} &= \widehat{\mathbf{A}} - \frac{2}{d}\big(\mathbf{1}\widehat{\mathbf{A}}\big)_{\mathrm{sym}} + \frac{1}{d^2}\mathbf{1}\widehat{\mathbf{A}}\mathbf{1}
\end{aligned} \tag{6}$$

Here $\mathbf{1} \in \mathbb{R}^{d \times d}$ is a matrix each element of which is 1, $\mathbf{I}$ is the identity matrix, and $\circ$ indicates the Hadamard product. We denote $(\mathbf{U})_{\mathrm{sym}} = \frac{1}{2}(\mathbf{U} + \mathbf{U}^T)$. Hereafter, we use $\mathbf{A}_{\boldsymbol{\theta}}(\mathbf{z})$ to indicate that the BDC matrix is computed from the network parameterized by $\boldsymbol{\theta}$ with an input image $\mathbf{z}$.

As such, we formulate DeepBDC as a parameter-free, spatial pooling layer. It is highly modular, suitable for varying network architectures and for different frameworks of few-shot classification. The BDC matrix mainly involves standard matrix operations, appropriate for parallel computation on GPU. From Eq. (6), it is clear that the BDC matrix models non-linear relations among channels through Euclidean distance. The covariance matrices can be interpreted similarly, which, however, models linear relations among channels through inner product [49, Sec. 4.1]. Theoretically, they are quite different as the BDC matrices consider the joint distributions, while the covariance matrices only consider the marginal ones.



(a) Meta DeepBDC–Instantation with ProtoNet [33] as a blueprint.

(b) STL DeepBDC–Instantation based on Good-Embed [37] relying on non-episodic training.

Figure 1. Two instantiations of our DeepBDC for few-shot classification. Meta DeepBDC (a) is based on the idea of meta learning which depends on episodic training; here we take a 3-way 2-shot classification as an example for illustration. In STL DeepBDC (b), we train a network with a conventional softmax classifier and cross-entroy loss on the whole meta-training spanning all classes; during meta-testing, we use the trained network as an embedding model for feature extraction, constructing and training a logistic regression model for classification.

3.3. Instantiating DeepBDC for Few-shot Learning

We instantiate our DeepBDC based on meta-learning framework and on simple transfer learning framework, and the resulting Meta DeepBDC and STL DeepBDC are shown in Fig. 1a and Fig. 1b, respectively.

Meta DeepBDC Standard few-shot learning is performed in an episodic manner on a multitude of tasks. A task is often formulated as a N-way K-shot classification problem, which spans N classes each with K support images and Q query images, on a support set $\mathcal{D}^{\mathrm{sup}} = \{(\mathbf{z}_j, y_j)\}_{j=1}^{NK}$ and a query set $\mathcal{D}^{\mathrm{que}} = \{(\mathbf{z}_j, y_j)\}_{j=1}^{NQ}$. A learner is trained on $\mathcal{D}^{\mathrm{sup}}$ and makes predictions on $\mathcal{D}^{\mathrm{que}}$.

We instantiate Meta DeepBDC with ProtoNet [33] as a blue print. It learns a metric space where classification is performed by computing distances to the prototype of every class. On one task $(\mathcal{D}^{\mathrm{sup}}, \mathcal{D}^{\mathrm{que}})$, we feed image $\mathbf{z}_j$ to the network to produce the BDC matrix $\mathbf{A}_{\boldsymbol{\theta}}(\mathbf{z}_j)$. The prototype of the support class k is the average (Avg) of the BDC matrices belonging to its class:

$$\mathbf{P}_k = \frac{1}{K} \sum_{(\mathbf{z}_j, y_j) \in \mathcal{S}_k} \mathbf{A}_{\boldsymbol{\theta}}(\mathbf{z}_j) \tag{7}$$

where $\mathcal{S}_k$ is the set of examples in $\mathcal{D}^{\mathrm{sup}}$ labeled with class k. We produce a distribution over classes based on a softmax over distances to the prototypes of the support classes, and then formulate the following loss function:

$$\arg\min_{\boldsymbol{\theta}} \; - \sum_{(\mathbf{z}_j, y_j) \in \mathcal{D}^{\mathrm{que}}} \log \frac{\exp(\tau \mathrm{tr}(\mathbf{A}_{\boldsymbol{\theta}}(\mathbf{z}_j)^T \mathbf{P}_{y_j}))}{\sum_k \exp(\tau \mathrm{tr}(\mathbf{A}_{\boldsymbol{\theta}}(\mathbf{z}_j)^T \mathbf{P}_k))} \tag{8}$$

where τ is a learnable scaling parameter [5, 45, 46].

We train the learner by sampling tasks from a massive meta-training set $\mathcal{C}^{\mathrm{train}}$ where the number of classes is far larger than N. Then, we sample tasks from a held-out meta-testing set $\mathcal{C}^{\mathrm{test}}$ on which we evaluate the performance of the learner. The episodic training ensures consistency between meta-training and meta-testing, which is crucial for the meta-learning methods [33, 39].

STL DeepBDC This instantiation is based on a widely used simple transfer learning (STL) framework [10], in which a deep network is trained on a large dataset and is then used as an embedding model to extract features for downstream tasks with few labeled examples.

We train a conventional image classification task on the whole meta-training set $\mathcal{C}^{\mathrm{train}}$ spanning all classes. The cross-entropy loss between prediction and ground-truth labels is used for training a learner from scratch:

$$\arg\min_{\boldsymbol{\theta}, \mathbf{W}_k} \; - \sum_{(\mathbf{z}_j, y_j) \in \mathcal{C}^{\mathrm{train}}} \log \frac{\exp(\tau \mathrm{tr}(\mathbf{A}_{\boldsymbol{\theta}}(\mathbf{z}_j)^T \mathbf{W}_{y_j}))}{\sum_k \exp(\tau \mathrm{tr}(\mathbf{A}_{\boldsymbol{\theta}}(\mathbf{z}_j)^T \mathbf{W}_k))} \tag{9}$$

where $\mathbf{W}_k \in \mathbb{R}^{d \times d}$ is the k-th weight matrix and τ is a learnable scaling parameter. For a task $(\mathcal{D}^{\mathrm{sup}}, \mathcal{D}^{\mathrm{que}})$ sampled from meta-testing set $\mathcal{C}^{\mathrm{test}}$, we build and train a new linear classifier for K classes on $\mathcal{D}^{\mathrm{sup}}$, using the trained model as a feature extractor. Following [37], we adopt the logistic regression model for classification, and, instead of directly using the trained model for meta-testing tasks, a sequential self-distillation technique is used to distill knowledge from the trained model on the meta-training set.

By referring to Eq. (8) and Eq. (9), we can interpret $\mathbf{W}_k$ as the prototype of class k, a dummy BDC matrix learned through training. Note that similar interpretations are given in DeepEMD [47] and FRN [45]. It is worth mentioning that, by vectorization operation as described in Eq. (5) for both the BDC matrices and the weight matrices, the softmax function in Eq. (9) can be implemented via a standard fully-connected (FC) layer.

3.4. Relation with Previous Methods

Let $\{\mathbf{x}_j\}_{j=1}^n$ be features of a query image, viewed as the observations of a random vector X. One can compute the mean vector $\boldsymbol{\mu}_X = \frac{1}{n}\sum_{j=1}^n \mathbf{x}_j$, covariance matrix $\boldsymbol{\Sigma}_X = \frac{1}{n}\sum_{j=1}^n (\mathbf{x}_j - \boldsymbol{\mu}_X)(\mathbf{x}_j - \boldsymbol{\mu}_X)^T$ or Gaussian distribution $\mathcal{N}_{\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X}$, as image representations. Note that these representations have been extensively studied outside of the few-shot learning regime, where they are deemed global average pooling [13], bilinear [22] or covariance pooling [42], and Gaussian pooling [41], respectively. The corresponding prototypes of the support class, $\boldsymbol{\mu}_Y$, $\boldsymbol{\Sigma}_Y$ or $\mathcal{N}_{\boldsymbol{\mu}_Y, \boldsymbol{\Sigma}_Y}$, can be computed using the features of K support images.

ProtoNet [33] represents the images with the mean vector and measures the difference using Euclidean distance $\rho_{\mathrm{ProtoNet}}(X, Y) = \|\boldsymbol{\mu}_X - \boldsymbol{\mu}_Y\|^2$ or cosine similarity $\boldsymbol{\mu}_X^T \boldsymbol{\mu}_Y / (\|\boldsymbol{\mu}_X\| \|\boldsymbol{\mu}_Y\|)$ for metric learning.

CovNet [44] adopts the covariance matrices as image representations for improving the first-order representation. The covariance matrices are subject to signed square-root normalization and then are compared with the Euclidean distance in the matrix space (i.e., the Frobenius norm) $\rho_{\mathrm{CovNet}}(X, Y) = \|\boldsymbol{\Sigma}_X - \boldsymbol{\Sigma}_Y\|^2$.

ADM [20] proposes to use an asymmetric distribution measure (ADM) to evaluate the dis-similarity between the query image and the support class. The distributions of images are represented by multivariate Gaussians whose differences are measured by KL-divergence $\rho_{\mathrm{ADM}}(X, Y) = D_{\mathrm{KL}}(\mathcal{N}_{\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X} \| \mathcal{N}_{\boldsymbol{\mu}_Y, \boldsymbol{\Sigma}_Y})$.

DeepEMD [47] uses discrete distributions as image representations. Specifically, the discrete PDF of the query image is $f_X(\mathbf{x}) = \sum_{j=1}^n f_{\mathbf{x}_j} \delta_{\mathbf{x}, \mathbf{x}_j}$, where $f_{\mathbf{x}_j}$ denotes the probability of $\mathbf{x}_j$ and $\delta_{\mathbf{x}, \mathbf{x}_j}$ is the Kronecker delta which is equal to 1 if $\mathbf{x} = \mathbf{x}_j$ and zero otherwise. Let the PDF of a support image be $f_Y(\mathbf{y}) = \sum_{j=1}^n f_{\mathbf{y}_j} \delta_{\mathbf{y}, \mathbf{y}_j}$. The distance between $f_X(\mathbf{x})$ and $f_Y(\mathbf{y})$ is formulated as EMD, i.e., $\rho_{\mathrm{EMD}}(X, Y) = \min_{f_{\mathbf{x}_j, \mathbf{y}_l} \geq 0} \sum_{j=1}^n \sum_{l=1}^n f_{\mathbf{x}_j, \mathbf{y}_l} c_{\mathbf{x}_j, \mathbf{y}_l}$ with constraints $\sum_{l=1}^n f_{\mathbf{x}_j, \mathbf{y}_l} = f_{\mathbf{x}_j}$ and $\sum_{j=1}^n f_{\mathbf{x}_j, \mathbf{y}_l} = f_{\mathbf{y}_l}$ for $j, l = 1, \ldots, n$. Here $c_{\mathbf{x}_j, \mathbf{y}_l}$ is the transport cost. Thus, EMD seeks an optimal joint distribution $f_{XY}(\mathbf{x}_j, \mathbf{y}_l) \triangleq f_{\mathbf{x}_j, \mathbf{y}_l}$ such that the expectation of transportation cost is minimal [29, Sec. 2.3]. DeepEMD proposes a cross-reference mechanism to define $f_{\mathbf{x}_j}$ and $f_{\mathbf{y}_l}$, and a structured FC layer to handle K-shot classification ($K > 1$).

4. Experiments

We first describe briefly the experimental settings. Next, we perform ablation study on our two instantiations (i.e., Meta DeepBDC and STL DeepBDC) and make comparisons to the counterparts. Finally, we compare with state-of-the-art methods on six few-shot datasets, covering general object recognition, fine-grained categorization and cross-

d	Parameters (M)	1-shot		5-shot	
		Acc	Latency	Acc	Latency
1280	13.25	66.36±0.43	488	83.23±0.30	614
960	13.04	66.81±0.44	280	83.68±0.28	351
640	12.84	**67.34±0.43**	161	**84.46±0.28**	198
512	12.75	67.10±0.45	134	84.23±0.28	164
256	12.59	66.90±0.43	121	84.15±0.28	148
ProtoNet [33]		62.11±0.44	115	80.77±0.30	143

Similarity function	1-shot		5-shot	
	Acc	Latency	Acc	Latency
Inner product	**67.34±0.43**	161	82.38±0.32	193
Cosine similarity	61.74±0.42	172	82.49±0.31	207
Euclidean distance	56.70±0.45	163	**84.46±0.28**	198

(a) Meta DeepBDC based on ProtoNet [33] as a blueprint.

d	Parameters (M)	1-shot		5-shot	
		Acc	Latency	Acc	Latency
512	13.41	64.92±0.43	1110	84.61±0.29	2016
256	12.75	66.15±0.43	371	85.44±0.29	587
196	12.65	66.57±0.43	285	85.36±0.29	424
128	12.55	**67.83±0.43**	184	**85.45±0.30**	245
64	12.48	66.97±0.44	137	83.18±0.30	172
Good-Embed [37]		64.82±0.44	121	82.14±0.43	155

Classifier	1-shot		5-shot	
	Acc	Latency	Acc	Latency
Logistic regression	**67.83±0.43**	184	**85.45±0.30**	245
SVM	66.29±0.44	113	84.73±0.29	144
Softmax	66.30±0.44	1250	85.20±0.29	4374

(b) STL DeepBDC based on [37] relying on non-episodic training.

Table 2. Ablation analysis of our two instantiations of DeepBDC with the backbone of ResNet-12 on *mini*ImageNet. We report accuracy and latency (ms) of one meta-testing task for 5-way classification. Latency is measured with a GeForce GTX 1080.

Method	1-shot		5-shot	
	Acc	Latency	Acc	Latency
ProtoNet [33]	62.11±0.44	**115**	80.77±0.30	**143**
ADM [20]	65.87±0.43	199	82.05±0.29	221
CovNet [44]	64.59±0.45	120	82.02±0.29	144
DeepEMD [47]	65.91±0.82	457	82.41±0.56	12617
Meta DeepBDC	67.34±0.43	161	84.46±0.28	198
STL DeepBDC	**67.83**±0.43	184	**85.45±0.29**	245

Table 3. Comparison of accuracy and latency (ms) of 5-way classification to the counterparts with the backbone of ResNet-12 on *mini*ImageNet.

domain classification.

4.1. Experimental Settings

Datasets We experiment on two general object recognition benchmarks, i.e., *mini*ImageNet [39] and *tiered*ImageNet [31], and one fine-grained image classification dataset, i.e., CUB-200-2011 [40] (CUB for short). We also evaluate domain transfer ability of models by training on *mini*ImageNet and then test on CUB [40], Aircraft [24] and Cars [16].

Backbone network For fair comparisons with previous methods, we use two kinds of networks as backbones, i.e., ResNet-12 [18, 37] and ResNet-18 [1, 23, 34]. Same as commonly used practice, the input resolution of images is 84×84 for ResNet-12 and 224×224 for ResNet-18, respectively. Moreover, we adopt deeper models with higher capacity, i.e., ResNet-34 [13] with input images of 224×224 and a variant of ResNet-34 fit for input images of 84×84. Similar to [9, 20], we remove the last down-sampling of backbones to obtain more convolutional features.

Training Our Meta DeepBDC is based on meta-learning framework, depending on episodic training. Each episode (task) concerns standard 5-way 1-shot or 5-way 5-shot classification, uniformly sampled from meta-training or meta-testing set; following [5, 45, 47], before episodic training, we pre-train the models whose weights are used as initialization. Contrary to Meta DeepBDC, our STL DeepBDC is based on simple transfer learning framework, requiring non-episodic training. Following [37], we train a network as an embedding model with cross-entropy loss on the whole meta-training set spanning all classes; for each meta-testing task, we train a new logistic regression classifier using the features extracted by the embedding model.

In supplement (Supp.) S1, we provide statistics of datasets and the splits of meta training/validation/test sets, as well as details on network architectures, optimizers, hyperparameters, etc.

4.2. Ablation Study

We perform ablation analysis of our two instantiations and compare to the counterparts on *mini*ImageNet for 5-way task, with ResNet-12 as the backbone. Additional details on implementation of the counterparts and extra experiments are respectively given in Supp. S-2 and Supp. S-3.

Ablation analysis of Meta DeepBDC As the sizes of BDC matrices are quadratic in the number of channels, we introduce a 1×1 convolution (conv) layer, decreasing the channel number to d. In our implementation, each BDC matrix is vectorized as in Eq. (5), thus is of size $d(d+1)/2$. Tab. 2a (top) shows the effect of varying d on accuracy and on meta-testing time per episode. We can see that the highest accuracy is achieved when $d=640$; meanwhile, the meta-testing time only increases moderately as d enlarges. We also experiment by directly attaching BDC module to the backbone without the additional 1×1 conv layer; we achieve 67.10±0.43 and 84.50±0.28 for 1-shot and 5-shot, respectively, comparable to the best result obtained by using the additional 1×1 conv layer. Besides the inner product as depicted in Eq. (5), we can also use Euclidean distance or cosine similarity as metric, and the corresponding results are given in Tab. 2a (bottom). It can be seen that the inner product performs best for 1-shot task, while the Euclidean

Method	Backbone	***mini*ImageNet**		***tiered*ImageNet**	
		1-shot	5-shot	1-shot	5-shot
CTM [19]	ResNet-18	64.12±0.82	80.51±0.13	68.41±0.39	84.28±1.73
S2M2 [25]	ResNet-18	64.06±0.18	80.58±0.12	–	–
TADAM [26]	ResNet-12	58.50±0.30	76.70±0.38	–	–
MetaOptNet [18]	ResNet-12	62.64±0.44	78.63±0.46	65.99±0.72	81.56±0.63
DN4 [21] †	ResNet-12	64.73±0.44	79.85±0.31	–	–
Baseline++ [4] †	ResNet-12	60.56±0.45	77.40±0.34	–	–
Good-Embed [37]	ResNet-12	64.82±0.60	82.14±0.43	71.52±0.69	86.03±0.58
FEAT [46]	ResNet-12	66.78±0.20	82.05±0.14	70.80±0.23	84.79±0.16
Meta-Baseline [5]	ResNet-12	63.17±0.23	79.26± 0.17	68.62±0.27	83.29±0.18
MELR [11]	ResNet-12	67.40±0.43	83.40±0.28	72.14±0.51	87.01±0.35
FRN [45]	ResNet-12	66.45±0.19	82.83±0.13	71.16±0.22	86.01±0.15
IEPT [50]	ResNet-12	67.05±0.44	82.90±0.30	72.24±0.50	86.73±0.34
BML [51]	ResNet-12	67.04±0.63	83.63±0.29	68.99±0.50	85.49±0.34
ProtoNet [33] †	ResNet-12	62.11±0.44	80.77±0.30	68.31±0.51	83.85±0.36
ADM [20] †	ResNet-12	65.87±0.43	82.05±0.29	70.78±0.52	85.70±0.43
CovNet [44] †	ResNet-12	64.59±0.45	82.02±0.29	69.75±0.52	84.21±0.26
DeepEMD [47]	ResNet-12	65.91±0.82	82.41±0.56	71.16±0.87	86.03±0.58
Meta DeepBDC	ResNet-12	67.34±0.43	84.46±0.28	72.34±0.49	87.31±0.32
STL DeepBDC	ResNet-12	**67.83±0.43**	**85.45±0.29**	**73.82±0.47**	**89.00±0.30**

(a) Results on general object recognition datasets.

Method	Backbone	**CUB**	
		1-shot	5-shot
ProtoNet [33]	Conv4	64.42±0.48	81.82±0.35
FEAT [46]	Conv4	68.87±0.22	82.90±0.15
MELR [11]	Conv4	70.26±0.50	85.01±0.32
MVT [27]	ResNet-10	–	85.35±0.55
MatchNet [39]	ResNet-12	71.87±0.85	85.08±0.57
Wang *et al.* LR [43]	ResNet-12	76.16	90.32
MAML [12]	ResNet-18	68.42±1.07	83.47±0.62
Δ-encoder [32]	ResNet-18	69.80	82.60
Baseline++ [4]	ResNet-18	67.02±0.90	83.58±0.54
AA [1]	ResNet-18	74.22±1.09	88.65±0.55
Neg-Cosine [23]	ResNet-18	72.66±0.85	89.40±0.43
LaplacianShot [52]	ResNet-18	80.96	88.68
FRN [45] †	ResNet-18	82.55±0.19	92.98±0.10
Good-Embed [37] †	ResNet-18	77.92±0.46	89.94±0.26
ProtoNet [33] †	ResNet-18	80.90±0.43	89.81±0.23
ADM [20] †	ResNet-18	79.31±0.43	90.69±0.21
CovNet [44] †	ResNet-18	80.76±0.42	92.05±0.20
Meta DeepBDC	ResNet-18	83.55±0.40	93.82±0.17
STL DeepBDC	ResNet-18	**84.01±0.42**	**94.02±0.24**

(b) Results on fine-grained categorization dataset.

Table 4. Comparison with state-of-the-art methods for both general and fine-grained few-shot image classification. The best results are in **bold black** and second-best ones are in red. † Reproduced with our setting.

distance achieves the highest accuracy for 5-shot. The optimal setting achieved here is used throughout the remaining paper. Finally, we note that Meta DeepBDC has much better performance than the baseline (i.e., ProtoNet), regardless of the value of d, while increase of latency is small.

Ablation analysis of STL DeepBDC For each meta-testing task of STL DeepBDC, we need to build and train a new linear classifier which introduces parameters and computations. As the size of BDC matrix is quadratic in d, the number of parameters is considerable relative to that of training examples, particularly for larger d. Therefore, with increase of d, there exist greater risk of overfitting, which may explain why overall the accuracy becomes lower when d is larger for both 1-shot and 5-shot tasks, as shown in Tab. 2b (top). STL DeepBDC with $d=128$ obtains the best result, higher than the best result of Meta DeepBDC while taking comparable time. Besides the logistic regression model, we compare with softmax classifier and linear SVM as well. From the results in Tab. 2b (bottom), we can see that the softmax classifier is on par with SVM, while both of them are inferior to the logistic regression; the latency of logistic regression is larger than SVM while the softmax classifier takes remarkably larger time than the other two. At last, we mention that STL DeepBDC with $d=128$ outperforms the baseline of Good-Embed by a large margin with moderate increase of latency.

The i.i.d. assumption underlying DeepBDC The BDC metric depends on the i.i.d. assumption [35] which is common in statistics and machine learning. As in Sec. 3.2, by viewing each channel (feature map) as a random observation, we obtain $d\times d$ matrices corresponding to the spatial pooling. Alternatively, one can regard each spatial feature as a random observation, leading to $hw\times hw$ matrices which correspond to a channel pooling; however, for 1-shot/5-shot task and with the same setting as in Tab. 2a, this produces 62.55±0.45/78.88±0.32 and 63.95±0.45/79.45±0.32 for Meta DeepBDC ($d=640$) and STL DeepBDC ($d=128$), respectively, much lower than the accuracies of spatial pooling. Note that the i.i.d. assumption may not hold for either spatial or channel pooling; our comparison suggests the spatial pooling is a better option.

Comparison to the counterparts Here we compare with the counterparts whose representations are based on distribution modeling. Like our DeepBDC, both ADM and CovNet need to estimate second moments, which leads to quadratic increase of representations. Therefore, for a fair comparison with them, we also add a 1×1 convolution with d channels for dimension reduction, obtaining the best results for them by tuning d. The comparion results are presented in Tab. 3.

Regarding the *accuracy*, we have several observations. (1) ProtoNet is inferior to CovNet and ADM, suggesting that second moments have better capability to model marginal distributions than first moment. (2) DeepEMD outperforms CovNet and ADM, which indicates that joint distribution modeling via EMD is superior to modeling of marginal distributions. (3) Both our two instantiations outperform the counterparts by large margins. We attribute this to that BDC has stronger capability of statistical dependency modeling by effectively harnessing joint distributions. As to the *latency*, our two instantiations both take a little longer time than ProtoNet and CovNet, while being

Method	Backbone	5-shot
Baseline [4]	ResNet-18	65.57±0.70
Baseline++ [4]	ResNet-18	62.04±0.76
GNN+FT [38]	ResNet-12	66.98±0.68
BML [51]	ResNet-12	72.42±0.54
FRN [45]	ResNet-12	77.09±0.15
ProtoNet [33] †	ResNet-12	67.19±0.38
Good-Embed [37] †	ResNet-12	67.43±0.44
ADM [20] †	ResNet-12	70.55±0.43
CovNet [44] †	ResNet-12	76.77±0.34
Meta DeepBDC	ResNet-12	77.87±0.33
STL DeepBDC	ResNet-12	**80.16±0.38**

(a) *mini*ImageNet → CUB.

Method	Backbone	5-shot
ProtoNet [33] †	ResNet-12	55.96±0.38
ADM [20] †	ResNet-12	65.40±0.36
CovNet [44] †	ResNet-12	63.56±0.37
Baseline [4] †	ResNet-12	59.04±0.36
Baseline++ [4] †	ResNet-12	56.50±0.38
Good-Embed [37] †	ResNet-12	58.95±0.38
Meta DeepBDC	ResNet-12	68.67±0.39
STL DeepBDC	ResNet-12	**69.07±0.39**

(b) *mini*ImageNet → Aircraft.

Method	Backbone	5-shot
ProtoNet [33] †	ResNet-12	46.30±0.36
ADM [20] †	ResNet-12	53.94±0.35
CovNet [44] †	ResNet-12	52.90±0.37
Baseline [4] †	ResNet-12	50.29±0.37
Baseline++ [4] †	ResNet-12	46.44±0.37
Good-Embed [37] †	ResNet-12	50.18±0.37
Meta DeepBDC	ResNet-12	54.61±0.37
STL DeepBDC	ResNet-12	**58.09±0.36**

(c) *mini*ImageNet → Cars.

Table 5. Comparison with state-of-the-art methods for 5-way 5-shot classification in cross-domain scenarios. The best results are in **bold black** and second-best ones are in red. † Reproduced with our setting.

comparable to ADM. Notably, DeepEMD is computationally expensive, ∼ 2 times and 50 times slower than the other methods for 1-shot and 5-shot tasks, respectively.

4.3. Comparison with State-of-the-art Methods

General object recognition According to Tab. 4a, on *mini*ImageNet, for 1-shot task Meta DeepBDC is on par with state-of-the-art MELR while STL DeepBDC is better than it; for 5-shot task, Meta DeepBDC and STL DeepBDC outperform BML, which previously achieved the best result, by 0.83 percentage points (abbreviated as pp hereafter) and 1.82 pp, respectively. Our Meta DeepBDC can be further improved by combining Image-to-Class Measure (DN4) following the idea introduced in [20]; accordingly, we achieve 67.86±0.41/85.14±0.29 for 1-shot/5-shot tasks, outperforming ADM+DN4 (66.53±0.43/82.61±0.30). On *tiered*ImageNet, for 1-shot task Meta DeepBDC is slightly better than state-of-the-art IEPT while STL DeepBDC outperforms it by 1.58 pp; for 5-shot task, Meta DeepBDC achieves slight gains (0.3 pp) over state-of-the-art MELR while STL DeepBDC has much larger gains (∼2.0 pp).

Fine-grained categorization Following [1, 4, 23], we conduct experiments on CUB with the original raw images. We reproduce ProtoNet and Good-Embed which are our baselines as well as FRN. From Tab. 4b, we can see that reproduced ProtoNet and Good-Embed are competitive with previous published accuracies, indicating our re-implemented baselines provide fair competition; moreover, our methods are high-ranking across the board, compared to state-of-the-art FRN, the gains of Meta DeepBDC and STL DeepBDC are 1.0/1.5 pp and 0.8/1.0 pp for 1-shot/5-shot task, respectively. Furthermore, by adopting ResNet-34 with 224×224 input images, our methods further improve. Specifically, Meta DeepBDC and STL DeepBDC achieve 85.25±0.39/94.31±0.17 and 84.69±0.43/94.33±0.21, respectively, outperforming corresponding baselines of ProtoNet (80.58±/90.11±0.26) and Good-Embed (79.33±0.48/90.10±0.28).

Cross-domain classification Finally, we evaluate 5-way 5-shot classification in cross-domain scenarios, by training on *mini*ImageNet and testing on three widely used fine-grained datasets. Except DeepEMD which is computationally prohibitive for us, we implement all counterparts based on distribution modeling, and Good-Embed on the three datasets, as well as Baseline and Baseline++ [4] on Aircraft and Cars. The results are shown in Tab. 5. On *mini*ImageNet→CUB, CovNet is very competitive, only slightly inferior to FRN, and both of them are much better than the other methods except ours. Meta DeepBDC and STL DeepBDC outperform the high-performing FRN by 0.8 pp and 3.1 pp, respectively. On *mini*ImageNet→Aircraft, our two instantiations improve over all the other compared methods by more than 3.2 pp. On *mini*ImageNet→Cars, ADM is superior among our competitors; compared to it, Meta DeepBDC and STL DeepBDC achieve 0.7 pp and 4.2 pp higher accuracies, respectively. These comparisons demonstrate that our models have stronger domain transfer capability.

5. Conclusion

In this paper, we propose a deep Brownian Distance Covariance (DeepBDC) method for few-shot classification. DeepBDC can effectively learn image representations by measuring, for the query and support images, the discrepancy between the joint distribution of their embedded features and product of the marginals. The core of DeepBDC is formulated as a modular and efficient layer, which can be flexibly inserted into deep networks, suitable not only for meta-learning framework based on episodic training, but also for the simple transfer learning framework that relies on non-episodic training. Extensive experiments have shown that our DeepBDC method performs much better than the counterparts, and furthermore, sets new state-of-the-art results on multiple general, fine-grained and cross-domain few-shot classification tasks. Our work shows great potential of BDC, a fundamental but overlooked technique, and encourages its future applications in deep learning.

References

[1] Arman Afrasiyabi, Jean-François Lalonde, and Christian Gagné. Associative alignment for few-shot image classification. In *ECCV*, 2020. 6, 7, 8

[2] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *ICML*, 2018. 2

[3] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 2

[4] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *ICLR*, 2019. 3, 7, 8

[5] Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, and Xiaolong Wang. Meta-baseline: Exploring simple meta-learning for few-shot learning. In *CVPR*, 2021. 3, 5, 6, 7

[6] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, 2005. 1

[7] Benjamin Cowley, Joao Semedo, Amin Zandvakili, Matthew Smith, Adam Kohn, and Byron Yu. Distance Covariance Analysis. In *AISTATS*, pages 242–251, 2017. 3

[8] Guneet S. Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *ICLR*, 2020. 3

[9] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. In *NIPS*, 2020. 2, 6

[10] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 5

[11] Nanyi Fei, Zhiwu Lu, Tao Xiang, and Songfang Huang. MELR: Meta-learning via modeling episode-level relationships for few-shot learning. In *ICLR*, 2021. 7

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 1, 3, 7

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 6

[14] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019. 2

[15] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015. 1

[16] Jonathan Krause, Michael Stark, Deng Jia, and Fei Fei Li. 3D Object representations for fine-grained categorization. In *ICCV Workshop*, 2013. 6

[17] Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. One shot learning of simple visual concepts. In *CogSci*, 2011. 1

[18] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019. 6, 7

[19] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding Task-Relevant Features for Few-Shot Learning by Category Traversal. In *CVPR*, 2019. 7

[20] Wenbin Li, Lei Wang, Jing Huo, Yinghuan Shi, Yang Gao, and Jiebo Luo. Asymmetric distribution measure for few-shot learning. *IJCAI*, 2020. 1, 2, 5, 6, 7, 8

[21] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *CVPR*, 2019. 7

[22] Tsung-Yu Lin, Aruni Roy Chowdhury, and Subhransu Maji. Bilinear convolutional neural networks for fine-grained visual recognition. *IEEE TPAMI*, 40(6):1309–1322, 2018. 5

[23] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *ECCV*, 2020. 6, 7, 8

[24] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 6

[25] Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *WACV*, 2020. 7

[26] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: task dependent adaptive metric for improved few-shot learning. In *NIPS*, 2018. 1, 7

[27] Seong-Jin Park, Seungju Han, Ji-Won Baek, Insoo Kim, Juhwan Song, Hae Beom Lee, Jae-Joon Han, and Sung Ju Hwang. Meta variance transfer: Learning to augment from the others. In *ICML*, 2020. 7

[28] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE TPAMI*, 27(8):1226–1238, 2005. 2

[29] Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019. 1, 5

[30] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In *ICLR*, 2020. 3

[31] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018. 6

[32] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Rogério Schmidt Feris, Abhishek Kumar, Raja Giryes, and Alexander M. Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *NIPS*, 2018. 7

[33] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017. 1, 2, 3, 4, 5, 6, 7, 8

[34] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip Torr, and Timothy Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018. 3, 6

[35] Gábor J. Székely and Maria L. Rizzo. Brownian distance covariance. *Annals of Applied Statistics*, 3:1236–1265, 2009. 2, 3, 7

[36] Gábor J. Székely, Maria L. Rizzo, and Nail K. Bakirov. Measuring and testing dependence by correlation of distances. *Annals of Statistics*, 35:2769–2794, 2007. 2, 3

[37] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *ECCV*, 2020. 3, 4, 5, 6, 7, 8

[38] Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation. *arXiv preprint arXiv:2001.08735*, 2020. 8

[39] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016. 1, 3, 5, 6, 7

[40] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 6

[41] Qilong Wang, Peihua Li, and Lei Zhang. G2DeNet: Global Gaussian distribution embedding network and its application to visual recognition. In *CVPR*, 2017. 5

[42] Qilong Wang, Jiangtao Xie, Wangmeng Zuo, Lei Zhang, and Peihua Li. Deep CNNs meet global covariance pooling: Better representation and generalization. *IEEE TPAMI*, 43(8):2582–2597, 2021. 5

[43] Yikai Wang, Chengming Xu, Chen Liu, Li Zhang, and Yanwei Fu. Instance Credibility Inference for Few-Shot Learning. In *CVPR*, 2020. 7

[44] Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *CVPR*, 2019. 1, 2, 5, 6, 7, 8

[45] Davis Wertheimer, Luming Tang, and Bharath Hariharan. Few-shot classification with feature map reconstruction networks. In *CVPR*, 2021. 1, 2, 3, 5, 6, 7, 8

[46] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *CVPR*, 2020. 2, 5, 7

[47] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. DeepEMD: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *CVPR*, 2020. 1, 2, 3, 5, 6, 7

[48] Hongguang Zhang and Piotr Koniusz. Power normalizing second-order similarity network for few-shot learning. In *WACV*, 2019. 3

[49] Jianjia Zhang, Lei Wang, Luping Zhou, and Wanqing Li. Beyond covariance: SICE and kernel based visual feature representation. *IJCV*, 129(2):300–320, 2021. 4

[50] Manli Zhang, Jianhong Zhang, Zhiwu Lu, Tao Xiang, Mingyu Ding, and Songfang Huang. IEPT: Instance-level and episode-level pretext tasks for few-shot learning. In *ICLR*, 2021. 7

[51] Ziqi Zhou, Xi Qiu, Jiangtao Xie, Jianan Wu, and Chi Zhang. Binocular mutual learning for improving few-shot classification. In *ICCV*, 2021. 7, 8

[52] Imtiaz Masud Ziko, Jose Dolz, Eric Granger, and Ismail Ben Ayed. Laplacian regularized few-shot learning. In *ICML*, 2020. 7