LibFewShot: A Comprehensive Library for Few-Shot Learning

Wenbin Li, Ziyi Wang, Xuesong Yang, Chuanqi Dong, Pinzhuo Tian, Tiexin Qin, Jing Huo, Yinghuan Shi, Lei Wang, Senior Member, IEEE, Yang Gao, and Jiebo Luo, Fellow, IEEE

Abstract—Few-shot learning, especially few-shot image classification, has received increasing attention and witnessed significant advances in recent years. Some recent studies implicitly show that many generic techniques or “tricks”, such as data augmentation, pre-training, knowledge distillation, and self-supervision, may greatly boost the performance of a few-shot learning method. Moreover, different works may employ different software platforms, backbone architectures and input image sizes, making fair comparisons difficult and practitioners struggle with reproducibility. To address these situations, we propose a comprehensive library for few-shot learning (LibFewShot) by re-implementing eighteen state-of-the-art few-shot learning methods in a unified framework with the same single codebase in PyTorch. Furthermore, based on LibFewShot, we provide comprehensive evaluations on multiple benchmarks with various backbone architectures to evaluate common pitfalls and effects of different training tricks. In addition, with respect to the recent doubts on the necessity of meta- or episodic-training mechanism, our evaluation results confirm that such a mechanism is still necessary especially when combined with pre-training. We hope our work can not only lower the barriers for beginners to enter the area of few-shot learning but also elucidate the effects of nontrivial tricks to facilitate intrinsic research on few-shot learning.

Index Terms—Fair comparison, few-shot learning, image classification, unified framework.

I. INTRODUCTION

FEW-SHOT LEARNING (FSL), especially few-shot image classification, has received considerable attention in recent years [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. It tries to learn an effective classification model from a few labeled training examples. A wide variety of advanced FSL methods has been proposed and significantly improved the classification performance on multiple benchmark datasets [11], [12], [13], [14], [15], [16], [17], [18].

Because of the extreme scarcity of training examples per class, it is almost impossible to only use the few available examples to learn an effective classifier to solve the few-shot classification problem. Therefore, the current FSL methods generally follow a paradigm of transfer learning, i.e., using a large labeled but class-disjoint auxiliary set to learn transferable knowledge (or representations) to boost the target few-shot task. More importantly, different from standard transfer learning [19], the existing FSL methods normally adopt a meta-training [20] or episodic-training mechanism [2] to train a few-shot model by constructing massive few-shot tasks (episodes) from the auxiliary set to simulate the target few-shot task. Like the target few-shot task, each simulated task also consists of a labeled support (training) set and an unlabeled query (test) set.

In general, these typical FSL methods can be roughly divided into two types, i.e., meta-learning based [20], [21] and metric-learning based [22], [23]. The former normally adopts a meta-learning or learning-to-learn paradigm [24], [25] to learn some kind of cross-task knowledge through an alternate optimization between the meta-learner and base-learner. In this way, it can make the model quickly generalize to new unseen tasks with a few training examples (i.e., test time fine-tuning). In contrast, the latter employs a learning-to-compare paradigm [2] to learn representations that can be transferred between tasks without test time fine-tuning (test-tuning for short). This is implemented by directly comparing the relations between query images and support images in each training task through an episodic training mechanism [2]. Both types of methods have greatly advanced the development of few-shot learning.

However, the question “Is meta- or episodic-training paradigm really crucial and optimal for the FSL problem?” has been raised recently by the community. Some recent works [26], [27], [28], [29], [30] have attempted to answer this question. For example, [27] finds that a simple baseline method, i.e., pre-training + test-tuning (called non-episodic based methods),...
can obtain competitive results when compared with the meta- or episodic-training paradigm based methods. Similarly, [30] proposes an improved baseline method by using logistic regression as the linear classifier in the test-tuning phase, which surprisingly achieves the state of the art. Therefore, can we conclude that the meta- or episodic-training paradigm is indeed NOT necessary for FSL?

In addition, we observe that the implementation and evaluation details of different FSL methods vary significantly. This could make fair comparison difficult, or even worse, make some conclusions questionable. Specifically, certain key discrepancies of existing FSL methods can be summarized as follows: 1) different software platforms (e.g., TensorFlow versus PyTorch); 2) different backbones (e.g., Wider ResNet12 versus ResNet12); 3) classifiers with different parameters (e.g., heavy-parametric classifier versus non-parametric classifier); 4) different input image sizes (e.g., $224 \times 224$ versus $84 \times 84$); 5) different test time evaluations (e.g., center crop evaluation versus raw evaluation). Clearly, such wide discrepancies are not amenable to a fair comparison and therefore cannot truthfully reflect the actual progress of FSL. Moreover, some FSL methods may employ additional fancy deep learning tricks, such as stronger data augmentation, knowledge distillation, self-supervision, label smoothing, and DropBlock, in the training process. The effects of different deep learning tricks are worth studying in FSL. Also, the additional used tricks are another key type of discrepancy.

Our Work: Therefore, to facilitate fair comparison and conveniently investigate the common issues in FSL, we develop a comprehensive library for few-shot learning (LibFewShot) by making most of the implementation details of different FSL methods consistent. To be specific, eighteen representative state-of-the-art FSL methods, including seven meta-learning based, six metric-learning based and five non-episodic based methods, are systematically re-implemented in a unified framework with the same single codebase in PyTorch. In LibFewShot, we try our best to ensure that all the methods use the same settings and the same bag of tricks, except for some specific tricks or specific neural architectures that are the main contributions of certain methods. In this way, we could take a true picture of the actual state-of-the-art results on FSL. More importantly, we are able to construct a large-scale study to analyze the impact of different tricks, such as pre-training, global classification, knowledge distillation and label smoothing, in a fair way.

Our Contributions: The main contributions of this work are as follows:

- We develop a unified framework LibFewShot with eighteen re-implemented FSL methods for the first time in the literature. It can be used as a toolbox and a platform to help practitioners efficiently use and reproduce FSL methods.
- We provide comprehensive evaluations of the eighteen methods on multiple benchmark datasets with various embedding backbones, by controlling the implementation details. This can reveal the actual progress of FSL and can be conveniently referenced to perform comparative experiments.
- We conduct a large-scale study on multiple representative FSL methods in a fair way, revealing that 1) pre-training indeed can learn a good initial representation but is not necessarily an optimal representation; 2) meta- or episodic-training can further improve this initial representation; 3) $\ell_2$ normalization of image feature vectors can significantly boost the final classification more than test-tuning in the test phase, especially in data-limited scenarios.
- We conduct comprehensive ablation studies for multiple deep learning tricks on the FSL problem, showing that many tricks could achieve significant algorithm-agnostic performance improvements and are universally applicable to different FSL methods.
- We have released LibFewShot as an open-source project on GitHub, and will continue to add new methods into this project. We welcome other researchers to contribute to this library to facilitate the community to conduct research on this important topic together.

II. OVERVIEW OF FEW-SHOT LEARNING METHODS

In this section, we will first introduce the problem formulation of FSL, and then review three kinds of FSL methods, i.e., non-episodic based methods, meta-learning based methods and metric-learning based methods, where multiple representative methods are further reviewed in detail.

A. Problem Formulation

In few-shot setting, there are usually three sets of data, including a target labeled support set $S$, a target unlabeled query set $Q$ and a class-disjoint auxiliary set $A$. In particular, $S$ and $Q$ share the same label space, which corresponds to the training and test sets in generic classification, respectively. The concept of “few-shot” in fact comes from $S$, where there are $C$ classes but each class only has $K$ (e.g., 1 or 5) labeled samples. We call this kind of classification task a $C$-way $K$-shot task. Clearly, such a few labeled samples in each class make it almost impossible to train an effective classification model, no matter using deep neural networks or traditional machine learning algorithms. Therefore, one solution is to use $A$ to boost the learning on the target task (i.e., $S$ and $Q$). The good point is that, $A$ generally enjoys more classes and samples per class than $S$, while the challenge is that, $A$ has a disjoint label space from $S$ and even may have a large domain shift from $S$.

Therefore, the current FSL methods mainly focus on how to effectively learn transferable knowledge from $A$ for fast adaptation (e.g., meta-learning based and non-episodic based methods) or for good generalization (e.g., metric-learning based methods) on $S$ with a few labeled support examples. Note that, in experiment at study, given a dataset $D$, it will be divided into $D_{\text{train}}$, $D_{\text{val}}$ and $D_{\text{test}}$ for training, validation and test, respectively. Typically, $D_{\text{train}}$ will be taken as the auxiliary set $A$, and multiple evaluation few-shot tasks $T = (S, Q)$ will be formed by randomly sampling from $D_{\text{val}}$ and $D_{\text{test}}$, respectively.

Notation: Following the literature, the auxiliary set, i.e., the set of base classes, is denoted as $A = \{X_i, y_i\}_{i=1}^N$, with the image $X_i \in \mathbb{R}^{H \times W \times 3}$ and the one-hot labeling vector $y_i \in Y = \{1, 2, \ldots, C\}$. The training set is $S = \{X_i, y_i\}_{i=1}^N$. The unlabeled query set is $Q$, where each task only consists of a few unlabeled images $y_i \in \{1, 2, \ldots, C\}$. The test set is $T$.
{0, 1}^{C_{base}}, where $C_{base}$ is the total number of base classes. For a $C$-way $K$-shot task with $C$ novel classes, each class has $K$ support images and $M$ query images, and the support set and query set are represented as $S = \{S_1, \ldots, S_C\} = \{X_i, y_i\}_{i=1}^{C\times K}$ and $Q = \{Q_i\}_{i=1}^{CM}$, respectively. Specifically, $S_i = \{X_i, y_i\}_{i=1}^{K}$ contains $K$ images and is the $c$-th class in $S$. $C\times K$ means that there are $C \times K$ images in total in the support set in a $C$-way $K$-shot task. Also, $C\times M$ indicates that there are $C \times M$ unlabeled query images in total in the query set. Let $f_\theta(\cdot)$ and $g_\omega(\cdot)$ denote the convolutional neural network based embedding backbone and classifier, respectively. Also, $g_\omega(\cdot)$ can be integrated with $f_\theta(\cdot)$ into a same network and trained in an end-to-end manner. For generic classification and the base learner in meta-learning, the cost function is represented as $L(g_\omega(f_\theta(X)), y)$. As for the metric-learning based FSL, the cost function is represented as $L(g_\omega(f_\theta(X)|S), y)$.

**Episodic-Training and Meta-Training:** To learn an effective FSL model, meta-training [20] or episodic-training mechanism [2] is normally adopted at the training stage. Both meta- and episodic-training rely on a lot of simulation few-shot tasks, which are randomly constructed from the auxiliary set $A$. Each simulated task $T$ consists of two subsets, $A_S$ and $A_Q$, which are akin to $S$ and $Q$, respectively. Note that because the labels in each simulated task are randomly assigned according to the original real labels in the auxiliary set, we call such a kind of label as local-label. In contrast, if the real labels of the auxiliary set are directly used, we mean that global-label is used.

At each iteration, one simulated task (episode), i.e., $T = (A_S, A_Q)$, is adopted to train the current model. Conceptually, tens of thousands of tasks, i.e., $\{T^i = (A^i_S, A^i_Q)\}_{i=1}^{M} \in \rho(T)$, will be randomly sampled from a task distribution $\rho(T)$ to train this model. The core principle is that the training condition (i.e., the training task) must match the test condition (i.e., the target task) [2]. From the perspective of meta-learning, as more tasks are observed, the model can use the accumulated meta-knowledge to adapt its own bias according to the characteristics of each task [25], [31].

**B. Non-Episodic Based Methods**

As illustrated in Fig. 1(a), the non-episodic based methods [9], [27], [28], [30], [32], [33] generally follow the standard transfer learning procedure [19], consisting of two phases, i.e., pre-training with base classes and test-tuning with novel classes.

**Pre-training with base classes:** In this phase, the whole auxiliary set $A$ is used to train a $C_{base}$-class classifier by using the standard cross-entropy loss as below,

$$\Gamma = \text{arg} \min_{\theta, \omega} \sum_{i=1}^{N} L^{CE}(g_\omega(f_\theta(X_i)), y),$$

where $L^{CE}$ is the cross-entropy loss function.

**Test-tuning with novel classes:** Test-tuning is performed in the test phase. Specifically, for each specific novel task $T = (S, Q)$, a new $C$-class classifier will be re-learned based on $S$ every time. Basically, the pre-trained embedding parameter $\theta$ is fixed to avoid over-fitting, because there is limited labeled data in $S$. Once the novel classifier is learned, the labels of $Q$ can be predicted.

**Representative methods** include Baseline [27], Baseline++ [27], RFS-simple [27], SKD-GEN0 [32], S2M2 [33] and Neg-Cosine [34], etc. The main difference between the first three methods is that they use different classifiers at the test-tuning stage: (1) Baseline [27] adopts a linear layer, i.e., a fully-connected (FC) layer, as the new classifier; (2) Baseline++ [27] replaces the standard inner product (in the FC layer) with a cosine distance between the input feature and weight vector; (3) RFS-simple [27] employs logistic regression instead of the FC layer as the new classifier by first using $\ell_2$ normalization for the feature vector.

SKD-GEN0 [32] also uses logistic regression as the classifier as RFS-simple [27], where the only difference is that additional rotation-based self-supervision [35] is further introduced into the pre-training stage. In addition, both [27] and [32] develop an extended version, respectively, by using knowledge distillation [36]. As for S2M2, more auxiliary tasks, such as Manifold Mixup [37], rotation [35] and Exemplar [38], are introduced into the pre-training stage to learn more powerful representations. Neg-Cosine [34] introduces a negative margin loss, i.e., a negative-margin cosine softmax loss, at the pre-training stage, and shows that this will benefit the novel classes in the test-tuning phase.
Discussions: The above non-episodic based methods have achieved surprisingly good results with a much simpler methodology, shaking the foundation of the current pure meta- or episodic-training based methods. The question of “Should we discard meta- or episodic-training in FSL?” will be interesting to investigate. On the other hand, intuitively, the cross-entropy loss used in the pre-training stage may make the learned representations overfit the seen base classes, thus lacking generalization ability for unseen classes. Moreover, the non-episodic methods strictly follow the paradigm of standard transfer learning and heavily focus on improving the pre-training stage by utilizing the latest and popular deep learning tricks, which may somewhat overlook the intrinsic problems of FSL.

C. Meta-Learning Based Methods

As illustrated in Fig. 1(b), meta-learning based methods [3, 12, 26, 39, 40, 41, 42] normally perform a meta-training paradigm on a family of few-shot tasks constructed from the base classes at the training stage, aiming to make the learned model able to quickly adapt to unseen novel tasks at the test stage. In particular, the meta-training procedure consists of a two-step optimization between the base-learner and meta-learner. Specifically, given a sampled task $\mathcal{T} = \langle \mathcal{A}_S, \mathcal{A}_Q \rangle$, Step-1 (i.e., base-learning or inner loop) is to use $\mathcal{A}_S$ (i.e., training examples in each task) to learn the base-learner. Next, in Step-2 (i.e., meta-tuning or outer loop), $\mathcal{A}_Q$ (i.e., test samples in each task) is employed to optimize the meta-learner. In this way, the meta-learner is expected to learn a kind of across-task meta-knowledge, which can be used for the fast adaptation on novel tasks.

Model-Agnostic Meta-Learning (MAML) is one representative method [3], whose core idea is to train a model’s initial parameters by involving the second-order gradients, making this model able to rapidly adapt to a new task just with one or a few gradient steps. Specifically, in the base-learning phase (inner loop), given $\mathcal{T} = \langle \mathcal{A}_S, \mathcal{A}_Q \rangle$, the current model $F_\theta = f_\theta \circ g_\omega$ and $\Theta = \Theta_0$, we can obtain the $m$-th inner loop gradient update as,

$$\Theta_m = \Theta_{m-1} - \alpha \nabla \Theta_{m-1} \mathcal{L}_{\mathcal{A}_S}(F_{\Theta_{m-1}}),$$  \hspace{1cm} \text{(2)}

where $\Theta = \{\theta, \omega\}$, $\alpha$ is a step size hyper-parameter, and $m$ is the total number of inner iterations. Next, in the meta-tuning phase (outer loop), the parameter of the model is truly updated over the previous parameter $\Theta$ rather than $\Theta_m$ by using the query set $\mathcal{A}_Q$, i.e.,

$$\Theta = \Theta - \beta \nabla \Theta \mathcal{L}_{\mathcal{A}_Q}(F_{\Theta_m}),$$  \hspace{1cm} \text{(3)}

where $\beta$ is a meta-step size hyper-parameter.

Ridge Regression Differentiable Discriminator (R2D2) is designed from another perspective [26], by adopting a standard machine learning algorithm such as ridge regression as the base-learner classifier $g_\omega(\cdot)$ in the inner loop. Note that the base-learner classifier in MAML is a standard FC layer. The advantage of R2D2 is that ridge regression enjoys a closed-form solution, which can make the base-learning phase more efficient. Specifically, the ridge regression with parameter matrix $W \in \mathbb{R}^{d \times c}$ is formulated as,

$$\Gamma = \arg \min_{W} ||XW - Y||^2 + \lambda \|W\|^2$$

$$= (X^\top X + \lambda I)^{-1} X^\top Y,$$  \hspace{1cm} \text{(4)}

where $X \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^{n \times c}$ denote $n$ input samples with $d$-dimensional features and the corresponding labels (i.e., $c$ classes), respectively, $I$ is the identity matrix, and $\lambda$ is a regularization hyper-parameter.

Specifically, suppose there are $n$ support images in $\mathcal{A}_S$, and $f_\omega(\cdot)$ can be used to obtain the feature embeddings, i.e., $X = f_\omega(\mathcal{A}_S) \in \mathbb{R}^{n \times d}$. In the base-learning phase, the optimal parameter matrix $W^*$ can be easily obtained according to (4). Next, in the meta-tuning phase, the predictions of $X_Q = f_\omega(\mathcal{A}_Q)$ can be achieved as,

$$\hat{Y} = \alpha X_Q W^* + \beta,$$  \hspace{1cm} \text{(5)}

where $\alpha$ and $\beta$ are scale and bias, respectively, which can be learned by optimizing the meta-loss $\mathcal{L}(Y, \hat{Y})$.

Other Representative Methods include Latent Embedding Optimization (LEO) [40], Almost No Inner Loop (ANIL) [41], Body Only update in Inner Loop (BOIL) [43], MetaOptNet [12] and Versa [39]. Specifically, LEO, ANIL and BOIL all follow the same optimization procedure as MAML. The core idea of LEO is to optimize the meta-learning process within a low-dimensional latent space, and to learn a generative distribution of model parameters, instead of directly learning the explicit high-dimensional model parameters in MAML. ANIL tries to remove the inner loop updates for the embedding backbone (i.e., $f_\omega$), but only applies the inner loop adaptation to the classifier (i.e., $g_\omega$), which means that only $\omega$ is updated in (2). In contrast with ANIL, BOIL updates only the embedding backbone $f_\omega$ but freezes the update of the classifier $g_\omega$ in the inner loop. Similar to R2D2, MetaOptNet also attempts to use a convex base learner, i.e. linear support vector machine (SVM), in the inner loop for FSL. Different from the above four methods, Versa is designed from a new perspective of Bayesian learning by introducing a versatile amortization network.

Discussions: We can see that there are mainly two development directions in the meta-learning based methods: (1) an implicit two-loop optimization direction following the procedure of MAML, such as LEO, ANIL and BOIL; (2) an explicit two-loop optimization direction following the procedure of R2D2, e.g. MetaOpeNet. The former follows a trend of designing a more efficient optimization-based meta-learning method, aiming to address the complicated optimization problem of meta-training. The latter aims to design a more effective base learner by introducing the traditional and classic machine learning algorithms into the paradigm of meta learning. On the other hand, note that the early meta-learning based methods mainly employ the pure meta-training paradigm to learn a model from scratch. Some methods, such as MTL [13] and LEO [40], have already introduced pre-training into the training process, because the pre-training technique can be easily leveraged as pre-processing. Therefore, the effect of pre-training is worth further investigating in meta-learning based FSL methods.
D. Metric-Learning Based Methods

As illustrated in Fig. 1(c), different from the two-loop structure of meta-learning based methods, metric-learning based methods [4], [15], [22], [23], [44], [45], [46], [47] directly compare the similarities (or distances) between the query images and support classes (i.e., learning-to-compare) through one single feed-forward pass through the episodic-training mechanism [2]. In other words, for each input query image, the entire support set $A_S$ is jointly encoded into the latent embedding space simultaneously, and their relationships (i.e., outputs) are used to classify. In this way, i.e., by conditioning on the support set, it is able to enable the model adapt to the characteristics of each task, and make the learned representations transferable between different tasks.

Prototypical Networks (ProtoNet) is a typical metric-learning based method [23], which takes the mean vector of each support class as its corresponding prototype representation, and then compares the relationships between the query image and prototypes. Specifically, given a few-shot task $T = (A_S, A_Q)$, $A_S = \{S_1, S_2, \ldots, S_K\}$, the prototype $c_i \in \mathbb{R}^d$ of each class $S_i$ can be formulated as,

$$
c_i = \frac{1}{|S_i|} \sum_{j=1}^{K} f_\theta(X_j),
$$

(6)

where $X_j \in S_i$, $|S_i| = K$ denotes there are $K$ images (i.e., $K$-shot) in the i-th support class. Here, $f_\theta(X_j) \in \mathbb{R}^d$ means $f_\theta(\cdot)$ extracts $d$-dimensional global feature representation for each input image. Given a distance function $D(\cdot, \cdot)$, such as euclidean distance, the predicted posterior probability distributions of a query image $Q$ is,

$$
\rho(y = i|Q) = \frac{\exp\left(-D(f_\theta(Q), c_i)\right)}{\sum_{j=1}^{C} \exp\left(-D(f_\theta(Q), c_j)\right)}.
$$

(7)

Specifically, at the training stage, the standard cross-entropy loss can be employed to train the entire model. Also, during test, the nearest-neighbor classifier (1-NN) can be conveniently used for prediction.

Deep Nearest Neighbor Neural Network (DN4) is another representative method [15], which argues that performing pooling on local features into a compact global-level representation will lose considerable discriminative information. Instead, DN4 advocates to directly use the raw local features and employs a local descriptor based image-to-class (I2C) measure to learn transferable local features. Specifically, given an input image $X$, without the last pooling or FC layer of the embedding network, $f_\theta(X) \in \mathbb{R}^{d \times h \times w}$ will be a three-dimensional tensor, and can be reshaped as a set of $d$-dimensional local descriptors,

$$
f_\theta(X) = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n},
$$

(8)

where $x_i$ is the $i$-th local descriptor and $n = h \times w$ is the total number of local descriptors for image $X$. Note that, in ProtoNet, both query image and the images in the support classes are represented with a global feature vector, respectively. Especially, the class prototype is also an average-pooling of multiple global feature vectors (e.g., the $K$-shot setting). In contrast, in DN4, both query image and each support class are represented with a set of local descriptors without any pooling.

Suppose a query image $Q$ and a support class $S$ are represented as $f_\theta(Q) = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}$ and $f_\theta(S) = [f_\theta(X_1), \ldots, f_\theta(X_K)] \in \mathbb{R}^{d \times nK}$, respectively. The image-to-class measure will be calculated as

$$
T = \text{TopK}\left(\ell_2(f_\theta(Q))^T \cdot \ell_2(f_\theta(S))\right) \in \mathbb{R}^{n \times k},
$$

$$
D_{\text{I2C}}(Q, S) = \sum_{i=1}^{n} \sum_{j=1}^{k} T_{ij},
$$

where $\ell_2(\cdot)$ denotes applying the $\ell_2$-normalization on the first dimension of $f_\theta(S)$ or $f_\theta(Q)$, and Topk(·) means selecting the $k$ largest elements in each row of the correlation matrix between $Q$ and $S$.

Other Representative Methods include Relation Network (RelationNet) [4], Covariance Metric Network (CovaMNet) [14], Cross Attention Network (CAN) [48], Deep Earth Mover’s Distance (DeepEMD) [45], Few-shot Embedding Adaptation with Transformer (FEAT) [16], and Relational Embedding Network (RENet) [47], etc. The core of RelationNet is to learn a non-linear metric through a deep convolutional neural network, instead of choosing a specific metric function, e.g., euclidean distance. Instead of using traditional first-order class representations, e.g., mean vector, CovaMNet proposes a second-order local covariance representation to represent each class along with a new covariance metric. From the perspective of attention, CAN proposes to calculate the cross attention between each pair of class feature and query feature so as to learn more discriminative features. Similar to DN4, DeepEMD also uses the set of local descriptors as the representation for an image and employs the Earth Mover’s Distance to calculate a structural distance between dense local representations of two images. FEAT proposes to take a set-to-set transformation via a transformer layer to make the global instance embedding of support set become task-specific for better adaptation. Similar to CAN, RENet proposes a self-correlational representation module and a cross-correlational attention module to learn relational patterns within and between images, respectively.

Discussions: From the recent advances, we can see that the main trends in metric-learning based methods are in two folds: (1) how to effectively represent each image and each support class; and (2) how to design a more powerful metric function. Specifically, the research trends show that using local descriptor representations may be a good choice and designing a task-adaptive metric function is important. In addition, we notice that because there are generally no data-dependent parameters in the classifier (i.e., 1-NN classifier), the metric-learning based methods do not have the test-tuning procedure at the test stage. Therefore, is the paradigm employed by metric-learning based methods reasonable? In other words, is test-tuning really essential for the FSL problem?

III. EVALUATION SETTINGS AND LIBFEWSHOT

Datasets: Our main experiments are conducted on two benchmark datasets, i.e., miniImageNet [2] and tieredImageNet [49].
Moreover, we also evaluate the cross-domain generalization ability of each FSL method, three fine-grained benchmark datasets, i.e., Stanford Dogs [50], Stanford Cars [51] and CUB Birds-200-2011 [52]. Following the literature, each data set is split into training (auxiliary), validation and test sets, respectively. The details can be seen in Table I. Note that all the images in the above datasets are resized to a resolution of $84 \times 84$.

**Backbone Architectures:** Following the literature [12], [23], [27], we adopt three different embedding backbones from shallow to deep, i.e., Conv64F, ResNet12 and ResNet18. Specifically, Conv64F contains four convolutional blocks, each of which consists of a convolutional (Conv) layer, a batch-normalization (BN) layer, a ReLU/LeakyReLU layer and a max-pooling (MP) layer, where the numbers of filters of these blocks are $\{64, 64, 64, 64\}$. ResNet12 consists of four residual blocks, each of which further contains three convolutional blocks (each is built as Conv-BN-ReLU-MP) along with a skip connection layer, where the numbers of filters of these blocks are $\{64, 120, 320, 640\}$. ResNet18 is the standard architecture used in [53]. One important difference between ResNet12 and ResNet18 is that ResNet12 uses Dropblock [54] in each residual block, while ResNet18 does not. Note that, in Table II, the numbers of filters of ResNet12 is $\{64, 96, 128, 256\}$; Conv64F has five Conv blocks; and Conv64F uses additional low-level features.

**Evaluation Protocols:** Following the prior works [2], [4], [23], in this paper, we control the evaluation setting for all methods, evaluate them on 600 sampled tasks and repeat this process five times, i.e., a total of 3000 tasks. The top-1 mean accuracy as well as the 95% confidence intervals will be reported. In addition, early works [2], [4], [23] generally use raw evaluation, i.e., directly resizing the test image into $84 \times 84$ for evaluation, while the recent work [9], [16], [27] has tried single center crop evaluation like the generic image classification [53]. To make the comparison more fair and can adapt to future development, we follow the latest setting to use the single center crop evaluation.

**Bag of Tricks:** Many works in other fields show that trick matters, especially in deep learning [55], [56], so as to FSL. For example, some recent FSL works have introduced such “tricks”, such as knowledge distillation [30], [32], self-supervision [57], [58] and Mixup [33], into the FSL problem. Specifically, we empirically summarize some key training tricks in FSL as below: 1) using data augmentation at the training stage (Train-DA); 2) augmenting the support set multiple times at the test stage (Test-DA); 3) pre-training on the auxiliary set (Pre-train); 4) global classification of using the global labels of the auxiliary set (Global-label); 5) Larger Episode size, i.e., increasing the number of tasks at each iteration; 6) higher way-number or higher shot-number during the episodic training; 7) using knowledge distillation or self-distillation (KD); 8) using self-supervision (SS); 9) using Dropblock; 10) using label smoothing; and 11) using more learnable parameters.

**LibFewShot:** In the literature, many implementation details or “tricks” are only briefly mentioned or even overlooked in many FSL works. However, these non-trivial tricks may lead to significant algorithm-agnostic performance boost, which will make the comparison somewhat unfair and make some conclusions untenable. In this sense, it will not only make beginners struggle with the reproduction of other comparison methods, but also hinder them from developing their own methods. On the other hand, there are many interesting issues worth studying under a unified framework, including 1) the doubts on the necessity of meta- or episodic-training mechanism posted by the recent non-episodic based methods, 2) the effects of different deep learning tricks on the FSL problem, 3) the actual progress of FSL in the case of restricting deep learning tricks, 4) the effect of transformers on FSL.

Therefore, to address the above issues, we develop a comprehensive library for few-shot learning (LibFewShot) by re-implementing the state-of-the-art methods into the same framework and applying the same training tricks to the maximum extent. So far, eighteen representative methods have been investigated, including five non-episodic based methods, seven meta-learning based methods and six metric-learning based methods, where the details can be seen in Table II. The details of the architecture of LibFewShot are illustrated in the supplementary material, available online.

### IV. Experimental Results and Discussions

#### A. Reproduction Results

To validate the correctness of our re-implementation, we adopt the original settings of these methods and re-implement them with LibFewShot. Our re-implemented results and their originally reported results on miniImageNet are shown in Table II. Importantly, the original implementation details and tricks are also listed in detail, which can intuitively show the differences between different FSL methods on the implementation.

#### Table I

| Dataset        | miniImageNet | tieredImageNet | Stanford Dogs | Stanford Cars | CUB Birds-200-2011 |
|----------------|--------------|----------------|---------------|---------------|-------------------|
| $C_{\text{all}}$ | 100          | 608            | 120           | 196           | 200               |
| $C_{\text{train}}$ | 64           | 351            | 70            | 130           | 130               |
| $C_{\text{val}}$ | 16           | 97             | 20            | 17            | 20                |
| $C_{\text{test}}$ | 20           | 160            | 30            | 49            | 50                |
| $N_{\text{val}}$ | 60000        | 779165         | 20580         | 16185         | 11788             |
| $N_{\text{train}}$ | 38400       | 448695         | 12165         | 10766         | 7648              |
| $N_{\text{val}}$ | 9600        | 124261         | 3312          | 1384          | 1182              |
| $N_{\text{test}}$ | 12000       | 206209         | 5103          | 4025          | 2958              |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE II

REPRODUCTION RESULTS ON miniImageNet USING THE ORIGINAL PAPER SETTINGS. RESULTS ARE REPORTED WITH THE MEAN ACCURACY OVER 3000 5-WAY 1-SHOT AND 5-WAY 5-SHOT TEST TASKS, RESPECTIVELY. GLOBAL-LABEL INDICATES THAT THE GLOBAL LABELS OF THE AUXILIARY SET ARE USED FOR PRE-TRAINING OR ADDITIONAL GLOBAL CLASSIFICATION DURING TRAINING. LOCAL-LABEL MEANS THAT ONLY SPECIFIC LOCAL LABELS ARE USED IN THE EPISODIC-OR META-TRAINING PHASE. TEST-TUNE MEANS TEST-TUNING OF USING THE SUPPORT SET DURING TEST.

| Method         | Embed. | Image Size | Training Tricks | 5-way 1-shot | 5-way 5-shot |
|----------------|--------|------------|-----------------|--------------|--------------|
|                |        |            | Lr/Optimizer/Decay Global-label Local-label Test-Tune Test-DA KD SS Reported Ours Reported Ours |
| Baseline       | Conv64F  | 84          | 0.001/Adam/No   | ✓            | ✓            | 42.11 | 42.34 | 62.53 | 62.18 |
| Baseline++     | ResNet18 | 224         | 0.001/Adam/No   | ✓            | ✓            | 51.75 | 51.18 | 74.27 | 74.05 |
|                | Conv64F  | 0.001/Adam/No | ✓            | ✓            | 48.24 | 46.21 | 66.43 | 65.18 |
|                | ResNet18 | 0.001/Adam/No | ✓            | ✓            | 51.87 | 53.60 | 75.68 | 73.63 |
| RFS-simple     | ResNet12 | ✓          | 0.05/SGD/Step   | ✓            | ✓            | 62.02 | 62.80 | 79.64 | 79.57 |
| RFS-distill    | ResNet12 | ✓          | 0.05/SGD/Step   | ✓            | ✓            | 64.82 | 63.44 | 82.14 | 80.17 |
| SKD-GEN0       | ResNet12 | ✓          | 0.05/SGD/Step   | ✓            | ✓            | 65.93 | 66.45 | 83.15 | 83.43 |
| SKD-GEN1       | ResNet12 | ✓          | 0.05/SGD/Step   | ✓            | ✓            | 67.04 | 67.09 | 83.54 | 83.67 |
| Neg-Cosine     | ResNet12 | ✓          | 0.003/Adam/Cosine | ✓          | ✓            | 63.85 | 63.28 | 81.57 | 81.24 |
| MAML           | Conv32F  | ✓          | 0.001/Adam/Step  | ✓            | ✓            | 48.70 | 47.41 | 63.11 | 65.24 |
| Versa          | Conv64F† | ✓          | 1e^{-4}/Adam/Step | ✓            | ✓            | 53.40 | 51.92 | 67.37 | 66.26 |
| R2D2           | Conv64F  | ✓          | 0.001/Adam/Step  | ✓            | ✓            | 49.50 | 47.57 | 65.40 | 66.68 |
| ANIL           | Conv64F  | ✓          | 0.001/Adam/Step  | ✓            | ✓            | 51.80 | 55.53 | 68.40 | 70.79 |
| LEO            | WRN-28-10 | ✓          | 4e^{-4}/Adam/No  | ✓            | ✓            | 61.76 | 55.89 | 77.59 | 70.55 |
| BOIL           | Conv64F  | ✓          | 0.001/Adam/Step  | ✓            | ✓            | 49.61 | 48.60 | 66.45 | 64.39 |
| MTL            | ResNet12 | ✓          | 0.1/SGD/Step     | ✓            | ✓            | 60.20 | 60.20 | 74.30 | 75.86 |
| ProtoNet†      | Conv64F  | ✓          | 0.001/Adam/Step  | ✓            | ✓            | 46.14 | 46.30 | 65.77 | 66.24 |
| RelationNet    | Conv64F  | ✓          | 0.001/Adam/Step  | ✓            | ✓            | 50.44 | 51.75 | 65.32 | 66.77 |
| CovaMNet       | Conv64F  | ✓          | 0.001/Adam/Step  | ✓            | ✓            | 51.19 | 53.36 | 67.65 | 68.17 |
| DN4            | Conv64F  | ✓          | 0.001/Adam/Step  | ✓            | ✓            | 51.24 | 51.95 | 71.02 | 71.42 |
| CAN            | ResNet12 | ✓          | 0.1/SGD/Step     | ✓            | ✓            | 63.85 | 66.62 | 79.44 | 78.96 |
| RENet          | ResNet12 | ✓          | 0.1/SGD/MultiStep | ✓          | ✓            | 67.60 | 66.83 | 82.58 | 82.13 |

Test-DA denote data augmentation during test. KD means knowledge distillation and SS means self-supervision. † and ‡ indicate that the standard backbones are different or slightly modified. - means the result is not reported in the original paper.

From Table II, we can observe that: 1) different FSL methods employ different backbones, especially the early meta- and metric-learning based methods only use a shallow backbone of Conv64F, making the comparison between different methods of using different backbones somewhat unfair; 2) non-episodic based methods, especially RFS [30] and SKD [32], indeed obtain the state-of-the-art results, but they employ a much deeper backbone and much more tricks than other methods; 3) both knowledge distillation (KD) and self-supervision (SS) can significantly boost the performance; 4) when using additional pre-training or global classification with global labels, MTL [13], CAN [48] and RENet [47] can achieve much better results. Notably, RENet consistently outperforms RFS-simple and RFS-distill no matter the officially reported results or our re-implemented results in both the 1-shot and 5-shot settings, which shows great potential of the paradigm of pre-training + meta-training.

B. Overview of the State of the Art

As seen in Table II, the implementation details of different methods vary a lot, which cannot trustingly reflect the actual progress of FSL. To this end, we keep most of the common
training tricks consistent for all the methods except some special tricks (e.g., self-supervision (SS) for SKD-GEN0 [32]) to make a relatively fair comparison. To be specific, we use the exactly same embedding backbone, fixed input image size, i.e., 84 × 84, the standard data augmentations (which consists of Resize, RandomCrop, RandomHorizontalFlip and ColorJitter) during training, and center crop evaluation for all the methods. As for the optimizer and learning rate scheduler, only minor modifications are made according to their settings in the original papers. All methods are trained with 100 epochs, except for Versa with 200 epochs. Also, 2000 episodes are used for miniImageNet per epoch, while 5000 episodes for tieredImageNet. The results are reported in Table III.

First, when using Conv64F as the embedding backbone, we can see that metric-learning based methods generally achieve relatively better results than meta-learning based and non-episodic based methods. This may be because when the embedding backbone is shallow, i.e., the feature representation is weak, the design or selection of metric function will be important. As also can be seen, the best methods of the metric-learning based and meta-learning based methods are RENet and LEO, respectively. One common characteristic of these two methods is that they employ both global- and local-label during training, which shows both global and local labels will benefit few-shot learning. In Section IV-D, we will further demonstrate this point. Similarly, MTL and CAN also employ such a kind of training strategy, both of which can obtain competitive results too. It is worth mentioning that some methods use more trainable parameters, such as Versa, LEO and MTL, which will also somewhat affect the final results.

Second, when using ResNet12 or ResNet18, a much deeper network, as the embedding backbone, we observe that the non-episodic based methods especially RFS-simple, Neg-Cosine and SKD-GEN0, perform significantly better than local-label based meta- and metric-learning methods, such as Versa, R2D2, ANIL, BOIL, ProtoNet, RelationNet, CovaMNet and DN4. However, when both global and local labels are used, we can see that MTL, CAN and RENet will be on par and even superior to RFS-simple and Neg-Cosine. Note that SKD-GEN0 utilizes an additional self-supervision trick, which essentially is not so fair for other methods. It is also worth noting that when using ResNet12 as the backbone most of these methods will perform much better than using ResNet18. This is because ResNet12 is much wider than ResNet18, i.e., ResNet12 enjoys more parameters than ResNet18. In addition, Dropblock is also used in ResNet12, while ResNet18 does not.

C. The Necessity of Episodic- and Meta-Training

One concern of our work is to investigate the necessity of episodic- and meta-training and try to answer the questions raised in Section II. To this end, we employ RFS-simple [30] without Test-DA as the benchmark, and use its pre-trained ResNet12 on miniImageNet as the embedding backbone.

Is test-tuning really important at the test stage in FSL? From Table III, we can see that RFS-simple consistently outperforms Baseline and Baseline++ when using ResNet12 or ResNet18 as the embedding backbone. However, the only difference between them is that RFS adopts logistic regression (LR) with $\ell_2$ normalization as the classifier in the test phase. That is to say, for each novel test task, a new LR classifier will be specially learned during test (i.e., test-tuning). Instead of using LR, we directly employ $\ell_2$ normalization based 1-NN for the final test classification, i.e., a non-parametric classifier without requiring test-tuning, and name this new variant as RFS-NN.

As seen in Table IV, without using $\ell_2$ normalization, the performance of RFS-simple will significantly degrade, especially in the 1-shot setting. In addition, using neither test-tuning nor $\ell_2$ normalization, RFS-NN still performs better than RFS-simple on the 5-shot tasks. More importantly, when only using $\ell_2$ normalization, RFS-NN could achieve very competitive results as RFS-simple on the 1-shot tasks and clearly outperforms RFS-simple on the 5-shot tasks especially in the higher way settings. This reveals that LR or test-tuning is not so important in the limited-data regime, but a good embedding and the $\ell_2$ normalization are! One reason is that $\ell_2$ normalization can align the distributions between the source and target domains, i.e., base and novel classes in FSL, to some extent to guarantee a good transferability. In other words, a misaligned-feature-norm phenomenon will weaken the transferability from the source domain to the target domain. Our observation is also corroborated by the finding in the work of [59] in the field of unsupervised domain adaptation.

In summary, we argue that test-tuning is not so necessary at the test stage in FSL because of the limited-data regime. In another word, a few labeled examples normally cannot learn an effective classifier in the test phase. This also demonstrates that the paradigm of metric-learning based methods is reasonable.

Is episodic- or meta-training necessary at the training stage in FSL? To answer this question, we employ three methods, i.e., ProtoNet, DN4 and R2D2, to episodic/meta training (fine-tuning) the same pre-trained feature embedding used in RFS-simple. That is to say, this is a kind of paradigm of using pre-training with global labels and meta fine-tuning with local labels. Note that for ProtoNet, we use a cosine distance instead of the original euclidean distance, because the above analysis shows that $\ell_2$ normalization is important. Specifically, for each method, we perform $C$-way 1-shot episodic/meta training and the corresponding $C'$-way 1-shot testing.

The results are shown in Fig. 2, whose details can be found in Section IV-D. As seen, for ProtoNet, DN4 and R2D2, all the episodic- or meta- fine-tuning (i.e., Two Stages) can further improve the performance over RFS-simple. Note that the data augmentation used in the fine-tuning phase (which only contains Resize and RandomResizedCrop) shall be somewhat different from the data augmentation used in the pre-training phase, which is one indetectable trick for this success overlooked in the literature. We also notice that if using somewhat different or more complicated metric functions in the meta fine-tuning phase will also significantly benefit the final performance. Overall, this simple experiment demonstrates that pre-training may obtain a good initial embedding, but it is not the optimal one. Therefore,
| Method          | Venue   | Embed. | Type          | Para. | Tricks | miniImageNet | tieredImageNet |
|----------------|---------|--------|---------------|-------|--------|--------------|----------------|
|                |         |        |               |       |        | 1-shot 5-shot 1-shot 5-shot |           |
| Baseline [27]  | ICLR’19 | Conv4F | Non-episodic  | 0.22M | ✓ ✓ ✓ ✓ | 44.90 63.96 48.20 68.96 |
| Baseline++ [27]| ICML’19 | Conv4F | Non-episodic  | 0.23M | ✓ ✓ ✓ ✓ | 48.54 65.47 49.73 70.14 |
| RFS-simple [30]| ECCV’20 | Conv4F | Non-episodic  | 0.22M | ✓ ✓ ✓ ✓ | 47.97 65.88 52.21 71.82 |
| Neg-Cosine [34]| ECCV’20 | Conv4F | Non-episodic  | 0.22M | ✓ ✓ ✓ ✓ | 47.34 65.97 51.21 71.57 |
| SKD-GENO [32]  | BMVC’21 | Conv4F | Non-episodic  | 0.22M | ✓ ✓ ✓ ✓ | 48.41 66.36 51.78 70.65 |
| MAML [3]       | ICML’17 | Conv4F | Meta          | 0.12M | ✓ ✓ ✓ ✓ | 49.55 64.92 50.98 67.12 |
| Versa [39]     | NeurIPS’18 | Conv4F | Meta          | 1.18M | ✓ ✓ ✓ ✓ | 52.75 67.40 52.28 69.41 |
| R2D2 [26]      | ICLR’19 | Conv4F | Meta          | 0.11M | ✓ ✓ ✓ ✓ | 51.19 67.29 52.18 69.19 |
| LEO [40]       | ICLR’19 | Conv4F | Meta          | 1.20M | ✓ ✓ ✓ ✓ | 53.31 67.47 58.15 74.21 |
| MTL [13]       | CVPR’19 | Conv4F | Meta          | 1.80M | ✓ ✓ ✓ ✓ | 46.70 64.79 49.11 69.13 |
| ANIL [41]      | ICLR’20 | Conv4F | Meta          | 0.12M | ✓ ✓ ✓ ✓ | 48.61 63.88 49.05 66.32 |
| BOIL [43]      | ICLR’21 | Conv4F | Meta          | 0.12M | ✓ ✓ ✓ ✓ | 47.92 64.39 50.04 65.51 |
| ProtoNet [23]  | NeurIPS’17 | Conv4F | Metric        | 0.11M | ✓ ✓ ✓ ✓ | 47.05 68.56 46.11 70.07 |
| RelationNet [4]| CVPR’18 | Conv4F | Metric        | 0.24M | ✓ ✓ ✓ ✓ | 51.52 66.76 54.37 71.93 |
| CovAMNet [14]  | AAAI’19 | Conv4F | Metric        | 0.11M | ✓ ✓ ✓ ✓ | 51.59 67.65 51.92 69.76 |
| DN4 [15]       | CVPR’19 | Conv4F | Metric        | 0.11M | ✓ ✓ ✓ ✓ | 54.47 72.15 56.07 75.75 |
| CAN [48]       | NeurIPS’19 | Conv4F | Metric        | 0.13M | ✓ ✓ ✓ ✓ | 55.88 70.98 55.96 70.52 |
| RENet [49]     | ICCV’21 | Conv4F | Metric        | 0.20M | ✓ ✓ ✓ ✓ | 57.62 74.14 61.62 76.74 |
|                |         |        |               |       |        | 1-shot 5-shot 1-shot 5-shot |           |
| Baseline [27]  | ICLR’19 | ResNet12 | Non-episodic | 12.47M | ✓ ✓ ✓ ✓ | 56.39 76.18 65.54 83.46 |
| Baseline++ [27]| ICML’19 | ResNet12 | Non-episodic | 12.47M | ✓ ✓ ✓ ✓ | 58.79 75.31 66.32 83.05 |
| RFS-simple [30]| ECCV’20 | ResNet12 | Non-episodic | 12.47M | ✓ ✓ ✓ ✓ | 61.65 78.88 70.55 84.74 |
| Neg-Cosine [34]| ECCV’20 | ResNet12 | Non-episodic | 12.47M | ✓ ✓ ✓ ✓ | 60.60 78.80 70.15 84.94 |
| SKD-GENO [32]  | BMVC’21 | ResNet12 | Non-episodic | 12.47M | ✓ ✓ ✓ ✓ | 66.40 83.06 71.90 86.20 |
| Versa [39]     | NeurIPS’18 | ResNet12 | Meta          | 13.25M | ✓ ✓ ✓ ✓ | 55.71 70.05 57.14 75.48 |
| R2D2 [26]      | ICLR’19 | ResNet12 | Meta          | 12.42M | ✓ ✓ ✓ ✓ | 59.52 74.61 65.07 83.04 |
| LEO [40]       | ICLR’19 | ResNet12 | Meta          | 12.60M | ✓ ✓ ✓ ✓ | 56.62 69.99 64.75 81.42 |
| MTL [13]       | CVPR’19 | ResNet12 | Meta          | 13.13M | ✓ ✓ ✓ ✓ | 62.67 79.16 68.68 84.58 |
| ANIL [41]      | ICLR’20 | ResNet12 | Meta          | 12.43M | ✓ ✓ ✓ ✓ | 52.77 68.11 55.65 73.52 |
| BOIL [43]      | ICLR’21 | ResNet12 | Meta          | 12.43M | ✓ ✓ ✓ ✓ | 58.87 72.88 64.66 80.38 |
| ProtoNet [23]  | NeurIPS’17 | ResNet12 | Metric        | 12.42M | ✓ ✓ ✓ ✓ | 57.10 74.20 62.93 83.30 |
| RelationNet [4]| CVPR’18 | ResNet12 | Metric        | 23.53M | ✓ ✓ ✓ ✓ | 55.22 69.25 56.86 74.66 |
| CovAMNet [14]  | AAAI’19 | ResNet12 | Metric        | 12.42M | ✓ ✓ ✓ ✓ | 54.69 70.72 56.03 75.21 |
| DN4 [15]       | CVPR’19 | ResNet12 | Metric        | 12.42M | ✓ ✓ ✓ ✓ | 59.14 75.26 64.41 82.59 |
| CAN [48]       | NeurIPS’19 | ResNet12 | Metric        | 12.65M | ✓ ✓ ✓ ✓ | 62.68 78.36 70.48 84.50 |
| RENet [49]     | ICCV’21 | ResNet12 | Metric        | 12.67M | ✓ ✓ ✓ ✓ | 64.81 79.90 70.14 82.70 |

Global-label indicates that the global labels of the auxiliary set are used for pre-training or global classification during training. Local-label means that only the specific local labels are used in the episodic- or meta-training phase. Test-tune means test-tuning of the support set at the test stage. Note that SKD-GENO uses an additional self-supervision trick that is the core of this method.
TABLE IV
NECESSITY OF TEST-TUNING IN THE TEST PHASE, WHERE RFS-SIMPLE IS TAKEN AS AN EXAMPLE METHOD BY USING DIFFERENT TEST SETTINGS

| Method   | Classifier | Test-tune | $\ell_2$ | (5,1) | (5,5) | (10,1) | (10,5) | (15,1) | (15,5) | (20,1) | (20,5) |
|----------|------------|-----------|---------|-------|-------|--------|--------|--------|--------|--------|--------|
| RFS-simple | LR         | ✓         | x      | 58.69 | 77.72 | 43.87  | 64.86  | 36.25  | 57.08  | 31.24  | 51.70  |
|           | LR         | ✓         | ✓      | 61.61 | 78.80 | 45.85  | 65.18  | 37.72  | 57.01  | 32.47  | 51.25  |
| RFS-NN   | 1-NN       | x         | ✓      | 60.64 | 76.85 | 44.89  | 66.20  | 36.86  | 58.46  | 31.70  | 53.15  |

Test-tune means test-tuning of using the support set at the test stage. $\ell_2$ means using $\ell_2$ normalization. $(C, K)$ means $C$-way $K$-shot tasks.

Fig. 2. Effects of using both global and local labels on miniImageNet with a ResNet12 backbone under both 5-way 1-shot and 5-shot settings. The 95% confidence intervals are shown in the form of error bars.

we argue that the episodic- or meta-training is worthy of further investigation at the training stage in FSL.

D. Ablation Study on Non-Trivial Tricks

In this section, we select multiple representative methods from the three kinds of FSL methods and conduct ablation studies on multiple non-trivial tricks. To be specific, miniImageNet and ResNet12 are taken as the default benchmark dataset and embedding backbone, respectively. Also, both 5-way 1-shot and 5-shot tasks are considered. Typically, non-trivial tricks, including global-label, local-label, strong data augmentation, knowledge distillation, label smoothing and self-supervision, are taken into consideration. In addition, RFS-simple, R2D2, ProtoNet, DN4, CAN-Local and RENet-Local will be selected as the representative FSL methods. Note that CAN-Local and RENet-Local are the variants of CAN and RENet of only using the local labels, respectively.

1) Effects of Global and Local Labels: To thoroughly investigate the effects of global and local labels, we consider three types of how to use both of them: 1) Pre-training with global labels at the first stage and episodic fine-tuning with local labels using a specific method at the second stage, we name this type as Two Stages for simplicity; 2) Global classifier with global labels + local classifier with local labels through episodic-training in a multi-task learning manner, we name this type as Global Classifier for short; 3) is similar to 2) but uses different data loaders for these two tasks, which is first adopted in RENet. We name the type of 3) as Auxiliary Dataloader for simplicity. Note that both 2) and 3) are one-stage methods, which use both global and local labels via a single training process. From the results in Fig. 2, we have the following observations.

1) All three types of using both global and local labels work well, and can effectively improve the performance over the vanilla version of only using the local or global labels. For example, under the 5-way 1-shot setting, ProtoNet+Two Stages (using both global and local labels) can achieve an accuracy of 63.56%, obtaining 6.46% improvements over the vanilla version (using local labels only), which is significantly better than RFS-simple (60.64%). Similarly, DN4+Two Stages (63.02%) and R2D2+Two Stages (62.04%) gains 4.34% and 2.52% improvements over their vanilla versions, respectively.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Knowledge distillation as an auxiliary task achieves accuracies of 63.27% 4) Effect of Knowledge Distillation: Knowledge distillation (KD) has been introduced for FSL by some recent FSL methods, such as RFS-distill and SKD-GEN1. To further verify its effect, we apply KD to three early FSL methods, i.e., ProtoNet, DN4 and R2D2. Following RFS-distill and SKD-GEN1, we also adopt the self-distillation to distill the knowledge from a trained few-shot model to a new identical model initialized from scratch for one.

As seen in Fig. 4, in both settings of 1-shot and 5-shot, KD can consistently improve the performance of the vanilla versions of all the three FSL methods. For example, under the 5-shot setting, the gained improvements for ProtoNet, DN4 and R2D2 are 1.86%, 1.73%, 1.94%, respectively. Similarly, under the 1-shot setting, KD boosts the performance of ProtoNet, DN4 and R2D2 by 2.18%, 0.65%, 2.06% improvements, respectively. This shows that KD is indeed effective in FSL. On the other hand, we shall notice that KD is a general trick, which is applicable to other FSL methods.

4) Effect of Self Supervision: Self-supervision methods have been shown to be effective in many fields of deep learning with the advantage of requiring no additional annotation costs. In the field of FSL, Gudarus et al. [62] has used the additional rotation prediction self-supervision [35] as an auxiliary parallel task. Similarly, SKD [32] also adopts the rotation prediction as an auxiliary task but in the pre-training stage. These works have demonstrated that self-supervision, especially rotation prediction self-supervision, is effective for FSL.

To further investigate whether self-supervision is a general trick for other FSL methods, we take the rotation prediction as an auxiliary task and employ ProtoNet, DN4 and R2D2 as three representative FSL methods. Specifically, all methods are

Fig. 3. Effects of using strong data augmentation (i.e., mixup and cutmix) for ProtoNet, R2D2 and RFS-simple on miniImageNet with ResNet12. The 95% confidence intervals are shown as error bars.

2) For the episodic-training based methods of only using local labels, such as ProtoNet, DN4 and R2D2, using extra global labels by pre-training (i.e., a two-stage way) is the most effective way. In contrast, for the methods originally designed with both global and local labels, such as CAN and RENet, using the global labels with a multi-task manner (i.e., a one-stage way) is generally the more effective way.

In summary, different from the recent doubts on the importance of local labels, the above observations reveal that both global and local labels are important for few-shot learning. This is because on one hand, the global labels indeed provide the model with good initial representations (i.e., priori knowledge), while it will make the model easily be biased to the base classes, weakening the model’s transferability to novel classes. On the other hand, the local labels used in the meta- or episodic-training mechanism can produce a task-level distribution consistency assumption by making the tasks sampled from both base and novel classes follow a similar task-level distribution. This will benefit the transferability of knowledge from based classes to the novel classes in a task-level perspective.

2) Effect of Strong Data Augmentation: Data augmentation is one of the most general and effective tricks in the field of deep learning. Therefore, it will be interesting to investigate the effect of strong data augmentation techniques for FSL. To be specific, we take mixup [60] with alpha of 0.2 and cutmix [61] with alpha of 1.0 (alpha is the hyper-parameter of the beta distribution in these two tricks) as the representative strong data augmentations and take ProtoNet, R2D2 and RFS-simple as the representative FSL benchmark methods.

From the results in Fig. 3, we can observe that: 1) cutmix can significantly improve the performance of all the three FSL methods. For example, in the 1-shot setting, cutmix obtains 2.24%, 3.75%, and 2.08% improvements over the vanilla versions of ProtoNet, R2D2 and RFS-simple, respectively. Similarly, in the 5-shot setting, cutmix obtains 3.69%, 5.71%, and 2.2% improvements over the three vanilla versions, respectively. 2) In most cases, mixup can effectively improve the performance of all the three FSL methods, especially for ProtoNet and R2D2. For example, in the 5-shot setting, mixup gains 2.57%, 2.54%, 0.41% improvements over the vanilla versions of ProtoNet, R2D2 and RFS-simple, respectively. 3) When using strong data augmentation, e.g., cutmix, R2D2 is surprisingly competitive to RFS-simple. That is to say, by just using a simple data augmentation technique, R2D2 + cutmix achieves accuracies of 63.27% and 80.32% in the 1-shot and 5-shot settings, respectively, which are already very competitive in Table III.

3) Effect of Knowledge Distillation: Knowledge distillation (KD) has been introduced for FSL by some recent FSL methods, such as RFS-distill and SKD-GEN1. To further verify its effect, we apply KD to three early FSL methods, i.e., ProtoNet, DN4 and R2D2. Following RFS-distill and SKD-GEN1, we also adopt the self-distillation to distill the knowledge from a trained few-shot model to a new identical model initialized from scratch for one.

As seen in Fig. 4, in both settings of 1-shot and 5-shot, KD can consistently improve the performance of the vanilla versions of all the three FSL methods. For example, under the 5-shot setting, the gained improvements for ProtoNet, DN4 and R2D2 are 1.86%, 1.73%, 1.94%, respectively. Similarly, under the 1-shot setting, KD boosts the performance of ProtoNet, DN4 and R2D2 by 2.18%, 0.65%, 2.06% improvements, respectively. This shows that KD is indeed effective in FSL. On the other hand, we shall notice that KD is a general trick, which is applicable to other FSL methods.
trained from scratch through the episodic-training mechanism by using an additional rotation classifier in a multi-task manner. The results are shown in Fig. 5. We can see that no matter in the 5-way 1-shot setting nor 5-way 5-shot setting, the rotation prediction self-supervision can consistently boost the performance of all three FSL methods. For example, in the 1-shot setting, the rotation self-supervision can boost ProtoNet, DN4 and R2D2 for 0.94%, 0.55%, 2.57% improvements, respectively. Similarly, in the 5-shot setting, the gained performance improvements of ProtoNet, DN4 and R2D2 are 3.27%, 1.39%, 1.17%, respectively. This reveals that the self-supervision task, especially the rotation prediction task, is indeed effective for FSL and can be a general trick for different FSL methods.

5) Effect of Label Smoothing: Label smoothing (LS) [63] attempts to prevent the model to be over-confident by softening the ground-truth labels, which is a common trick in the field of deep learning and representation learning. Therefore, it is also interesting to investigate its effect on the problem of FSL. To be specific, we apply LS to three FSL methods, including ProtoNet, R2D2 and RFS-simple. In addition, for each method of using LS, we adopt two different hyper-parameters for LS, i.e., 0.1 and 0.2, to balance the weight between the original ground-truth and the uniform distribution.

The results are reported in Fig. 6. From the results, not surprisingly, we can see that in most cases, LS can further improve the performance of the vanilla versions of the three FSL methods, especially in the 5-shot setting. For example, in the 5-shot setting, LS (0.2) boosts the performance of ProtoNet, R2D2 and RFS-simple by 1.05%, 1.90%, 1.31% improvements, respectively. This reveals that label smoothing can also be a general trick for boosting the performance of FSL methods.

V. USING TRANSFORMER IN FSL

In recent years, many new vision transformer based models have been proposed, such as ViT [64], DeiT [65], DETR [66], SETR [67], and Swin-Transformer [68], and shown promising results on a variety of vision tasks, compared with CNN architecture-based models. Therefore, it will be interesting to explore the effect of transformer in few-shot learning.

In particular, we implement four representative FSL methods, including ProtoNet, DN4, R2D2 and RFS, by using the tiny version of Swin Transformer (Swin-T for short) [68], a powerful transformer network, as the embedding backbone. Also, the versions using ResNet12 as the embedding backbone for these methods are taken as comparisons. Note that the common Swin-T is designed for the generic image classification with the image size of 224 × 224, while FSL normally uses an image size of 84 × 84. Therefore, for a fair comparison, we use an image size of 84 × 84 for Swin-T and change the down-sampling factors for each stage (linear embedding + transformer block) from (4, 2, 2, 2) to (3, 2, 2, 1) but keep all the other architecture hyper-parameters by default. In terms of data augmentation, RandomCrop, RandomHorizontalFlip, and ColorJitter, which are commonly used in few-shot learning, could not achieve good results when using Swin-T as the backbone. Therefore, we add RandAugment, mixup and cutmix into the data augmentation transformations for better results. In addition, we find that transformer is sensitive to the learning rate, so we follow the literature of generic classification, use AdamW as the optimizer, and carefully adjust each model’s learning rate.

The results are reported in Table V. From the results, we observe that: 1) In most cases, Swin-T shows worse results than ResNet12 on both miniImageNet and tieredImageNet. This is because transformers generally rely on a large-scale dataset for training to achieve excellent performance, because transformers lack locality and translation invariance properties that existed in the CNN architecture [69], [70]. 2) Swin-T can benefit from a large dataset, i.e., tieredImageNet. As seen, all methods especially DN4 and RFS, obtain much higher results on tieredImageNet than the results on miniImageNet under both 1-shot and 5-shot settings. 3) Interestingly, when using Swin-T, ProtoNet can achieve very competitive results with the results of using ResNet12. Also, DN4 with Swin-T achieves a higher result than DN4 with ResNet12 on tieredImageNet under the 1-shot setting.
TABLE V

Using Transformer in FSL on both miniImageNet and tieredImageNet. All models are trained from scratch without pre-training. The fourth column shows the total number of trainable parameters used by each model, and the fifth column denotes the details of optimization during training.

| Method    | Embedding | Image Size | Parameters | Optimizer / Lr / Decay | miniImageNet | tieredImageNet |
|-----------|-----------|------------|------------|------------------------|--------------|----------------|
|           |           |            | (5,1)      | (5,5)                  | (5,1)        | (5,5)          |
| ProtoNet  | ResNet12  | 84 x 84    | 12.42M     | Adam/1e-3/Step         | 57.10        | 62.93          | 83.30          |
|           | Swin-T    | 84 x 84    | 26.59M     | AdamW/5e-4/Cosine      | 57.23        | 62.33          | 81.83          |
| DN4       | ResNet12  | 84 x 84    | 12.42M     | Adam/1e-3/Step         | 59.14        | 64.41          | 82.59          |
|           | Swin-T    | 84 x 84    | 26.59M     | AdamW/5e-4/Cosine      | 54.29        | 64.94          | 79.36          |
| R2D2      | ResNet12  | 84 x 84    | 12.42M     | Adam/1e-3/Step         | 59.52        | 65.07          | 83.04          |
|           | Swin-T    | 84 x 84    | 26.59M     | AdamW/5e-4/Cosine      | 50.41        | 61.68          | 75.71          |
| RPS       | ResNet12  | 84 x 84    | 12.47M     | Adam/1e-3/Step         | 61.65        | 78.88          | 84.74          |
|           | Swin-T    | 84 x 84    | 26.64M     | AdamW/5e-4/Cosine      | 58.13        | 76.94          | 85.68          |

In summary, the above analyses reveal that transformers, as the latest popular and powerful architecture, are also promising and showing good potential in the field of few-shot learning. However, the key to achieving this expectation is how to effectively address the generic limitation, i.e., the data-hungry property, of transformers.

A. Cross-Domain Transferability

Cross-domain few-shot tasks have been introduced in many FSL works in the literature. Therefore, it will also be interesting to further evaluate the cross-domain transfer ability of different FSL methods in LibFewShot, which are not especially designed for this purpose. To this end, following the literature [27], we conduct an experiment on six cross-domain scenarios, e.g., miniImageNet → Stanford Dogs (mini->Dogs for short). In this experiment, all the few-shot models are trained on the source domain, e.g., miniImageNet or tieredImageNet, using a ResNet12 backbone with the same setting in Table III and directly tested on the target domain, e.g., Standford Dogs or CUB Birds.

The results are reported in Table VI. From the results, we have the following observations: 1) The models trained on miniImageNet and tieredImageNet can easily generalize to Stanford Dogs and CUB Birds that enjoy a small domain-shift. However, their performance significantly drops when performing on the Stanford Cars with a large domain-shift. 2) In most cases, especially in the large domain-shift scenario, i.e., mini→Cars, non-episodic based FSL methods perform somewhat better than metric-based or meta-based FSL methods. This means that pre-training is beneficial to cross-domain scenarios. Pre-training can easily bring prior strong representations which work well in natural image cross-domain scenarios. Especially, MTL, CAN and RENet use a pre-trained model or global labels, and they can also perform well on the mini→Cars task. 3) We can see that the classification accuracy on the cross-domain target datasets...
is significantly lower than that on the in-domain target datasets. This reveals that the current state-of-the-art FSL methods cannot handle the cross-domain scenarios well, which needs to be further investigated in the future.

VI. MAKING EARLY FSL METHODS GREAT AGAIN

We have conducted comprehensive ablation studies on multiple tricks in Section IV-D and demonstrated these tricks are non-trivial. This raises one interesting question, that is, whether we can make the early FSL methods great again using the non-trivial tricks? Recall that we have demonstrated that: 1) pre-training can provide good initial representations; 2) performing meta- or episodic-training (fine-tuning) over the pre-trained feature embedding can significantly benefit the final performance. Therefore, with the help of other useful tricks, “pre-training + episodic fine-tuning” will be a promising paradigm to make the early FSL methods great again. In this case, we will address two issues: 1) how to choose useful tricks in the pre-training phase, and 2) how to select helpful tricks in the episodic fine-tuning phase. To tackle the above two issues, we conduct two comprehensive ablation studies: 1) Ablation study on using different tricks at the pre-training stage on miniImageNet with a ResNet12 backbone, rotation, CutMix, Mixup, and label smoothing are combined with the generic classification task with the standard cross-entropy loss.
at the pre-training stage; 2) Ablation study on using different tricks at the episodic fine-tuning stage. Four tricks, including rotation, cutmix, mixup, and label smoothing, are considered at the pre-training stage. Also, three tricks, i.e., knowledge distillation, mixup and label smoothing, are concerned at the episodic fine-tuning stage. In addition, three early FSL methods, i.e., ProtoNet [23], DN4 [15] and R2D2 [26], are taken as the baselines.

Parameter settings: We conduct the pre-training experiment on the miniImageNet dataset with a ResNet12 backbone and train each model 100 epochs using the SGD optimizer with an initial learning rate of 0.05, a momentum of 0.9 and a weight decay of 0.0005. The learning rate is decayed by a factor of 0.1 at the 60th and 80th epoch, respectively. As for the hyper-parameters of the tricks, the loss weight for rotation is 2, the loss weight for knowledge distillation is 1, the alpha value for cutmix is 1.0, the alpha value for mixup is 0.2, and the parameter for label smoothing is 0.1. At the episodic fine-tuning stage, we fine-tune each model for 20 epochs using the SGD optimizer with the learning rate of 0.0001, momentum of 0.9, and weight decay of 0.9, where we do not apply any learning rate decay strategy. Note that both the temperature used for logits of the cross-entropy loss in fine-tuning and the temperature used for knowledge distillation are adjusted from 2 to 64 for different methods under different few-shot settings.

The experimental results at the pre-training stage are reported in Table VII. From the results, we have the following observations: 1) Combining all tricks together does not necessarily lead to the best performance. This is because some tricks may be mutually-exclusive or their functions somewhat overlap. Furthermore, because each trick has its own hyper-parameter, when multiple tricks are used together, the hyper-parameters are difficult to tune, making the optimization of the model be complicated. 2) Among the generic tricks (i.e., cutmix, mixup and label smoothing), cutmix alone can significantly improve the performance over the vanilla baseline. For example, when using ProtoNet as the classifier, cutmix achieves 64.01% and 81.43% classification accuracies under the 1-shot and 5-shot settings, respectively, which are 1.85% and 1.52% higher than the baseline without using any tricks. 3) As a self-supervision trick, rotation leads to the best results on all three FSL methods, when only a single trick is used. In addition, cutmix can further improve the performance over rotation in the 1-shot setting. Although mixup alone will decrease the performance, in some cases, mixup surprisingly can boost the performance by combining rotation and cutmix in the 5-shot setting for both ProtoNet and R2D2. 4) It is worth noting that the performance of DN4 is significantly lower than the other two methods. This is because during the pre-training stage, the classification model only learns the global features (after global average pooling), while DN4 is designed to perform classification using local features (before global average pooling) that are not explicitly optimized during pre-training.

The experimental results at the episodic fine-tuning stage are reported in Table VIII. Note that according to the results in Table VII, we take the pre-trained embedding backbone with both rotation and cutmix tricks as the initial embedding backbone at the episodic fine-tuning stage. Moreover, we take the three best FSL methods of using ResNet12 in Table III, i.e., SKD-GEN0 [32], MTL [13] and RENet [47], as the reference methods. From the results in Table VIII, we have the following observations: 1) All three early FSL methods can achieve competitive and even better results, compared with the latest methods, such as RENet and SKD-GEN0. For example, R2D2 obtains an accuracy of 67.71%, a new state of the art, gaining 5.04%, 2.3% and 1.31% improvements over MTL, RENet and SKD-GEN0 in the 1-shot setting, respectively. In addition, ProtoNet achieves an accuracy of 83.31% in the 5-shot setting, which is 4.15%, 3.41% and 0.25% higher than MTL, RENet and SKD-GEN0, respectively. 2) All three methods can significantly benefit from the pre-training stage, because pre-training does provide a good initial representation. Note that directly applying DN4 (focusing on local features) to the pre-trained embedding backbone (learning global features) without any fine-tuning, the performance will drop to some extent. However, the performance will be markedly improved after episodic fine-tuning is adopted. 3) Not all tricks are helpful at the episodic fine-tuning stage, because the performance at the pre-training stage is somewhat saturated. Fortunately, knowledge distillation generally can further enhance the final performance. Note that the reason why DN4 cannot benefit from knowledge distillation is that the pre-training just learned global features, while the global features are not suitable for applying knowledge distillation to local-feature-based DN4.

Overall, a combination of selected useful tricks indeed can make early FSL methods great again. On one hand, this reveals that the existing popular deep learning tricks are non-trivial, which can significantly boost the performance of FSL models. On the other hand, we may need to pay more attention to the essential problem of FSL, especially when we notice that these non-trivial tricks are universal to some extent.

VII. CONCLUSION

In this paper, we present a comprehensive library for few-shot learning (LibFewShot) by re-implementing the state-of-the-art FSL methods in a unified framework. Through LibFewShot, first, we are able to make a relatively fair comparison between different methods to reflect the actual progress of FSL. Second, we emphasize and demonstrate the necessity of episodic or meta-training. Third, we find that test-tuning is not very important at the test stage because of the limited-data setting in FSL, while a good embedding and $\ell_2$ normalization are truly important. Finally, we verify that many deep learning tricks are indeed non-trivial but are universal for different FSL methods. Also, we show that transformers are promising for FSL but are still needed to be further investigated in the future. We hope our work will facilitate healthy research on few-shot learning.
REFERENCES

[1] L. Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 4, pp. 594–611, Apr. 2006.

[2] O. Vinyals et al., “Matching networks for one shot learning,” in Proc. Neural Inf. Process. Syst., 2016, pp. 3630–3638.

[3] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 1126–1135.

[4] F. Sun, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, “Learning to compare: Relation network for few-shot learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 1199–1208.

[5] Q. Sun, Y. Liu, Z. Chen, T.-S. Chua, and B. Schiele, “Meta-transfer learning through hard tasks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 3, pp. 1443–1456, Mar. 2022.

[6] O. Vinyals et al., “Matching networks for one shot learning,” in Proc. Int. Conf. Learn. Representations, 2016, pp. 1–15.

[7] W. Chen, Y. Liu, Z. Kira, Y. F. Wang, and J. Huang, “A closer look at few-shot classification,” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–16.

[8] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto, “A baseline for few-shot image classification,” in Proc. Int. Conf. Learn. Representations, 2020, pp. 1–10.

[9] Y. Chen, X. Wang, Z. Liu, H. Xu, and T. Darrell, “A new meta-baseline for few-shot learning,” 2020, arXiv: 2003.04390.

[10] M. Abbas, Q. Xiao, L. Chen, P.-Y. Chen, and T. Chen, “Sharp-MAML: Sharpness-aware model-agnostic meta-learning,” in Proc. Int. Conf. Mach. Learn., 2022, pp. 10–32.

[11] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in Proc. Int. Conf. Learn. Representations, 2017, pp. 179–193.

[12] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, “Meta-learning with differentiable convex optimization,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 4135–4144.

[13] W. Li, L. Wang, J. Huo, Y. Shi, Y. Gao, and J. Luo, “Asymmetric distribution measure for few-shot learning,” in Proc. Int. Joint Conf. Artif. Intell., 2020, pp. 2957–2963.

[14] W. Li, J.-D.-C. Zhan, Y. Jiang, and Z.-H. Zhou, “Heterogeneous few-shot model rectification with semantic mapping,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 11, pp. 3878–3891, Nov. 2021.

[15] S. Yang, L. Liu, and M. Xu, “Free lunch for few-shot learning: Distribution calibration,” in Proc. Int. Conf. Learn. Representations, 2021, pp. 1–13.

[16] M. Abbas, Q. Xiao, L. Chen, P.-Y. Chen, and T. Chen, “Sharp-MAML: Sharpness-aware model-agnostic meta-learning,” in Proc. Int. Conf. Mach. Learn., 2022, pp. 10–32.

[17] S. Eidarian and N. Komodakis, “Dynamic few-shot visual learning without forgetting,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4367–4375.

[18] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, “Meta-learning with differentiable convex optimization,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 10657–10665.

[19] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, “Meta-transfer learning for few-shot learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 403–412.

[20] W. Li, J. Xu, J. Hua, L. Wang, Y. Gao, and J. Luo, “Distribution consistency based covariance metric networks for few-shot learning,” in Proc. AAAI Conf. Artif. Intell., 2019, pp. 8642–8649.

[21] W. Li, L. Wang, J. Xu, J. Hua, Y. Gao, and J. Luo, “Revisiting local descriptor based image-to-class measure for few-shot learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 7260–7268.

[22] H.-J. Ye, H. Hu, D.-C. Zhan, and P. Sha, “Few-shot learning via embedding adaptation with set-to-set functions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 8808–8817.

[23] J. Xie, F. Long, J. Lv, Q. Wang, and P. Li, “Joint distribution matters: Deep brownian distance covariance for few-shot classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2022, pp. 7972–7981.

[24] A. Afsarzadeh, Y. L. Archelle, J.-J. Lalonde, and C. Gagné, “Matching feature sets for few-shot image classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2022, pp. 9014–9024.

[25] S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[26] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, “Meta-learning with memory-augmented neural networks,” in Proc. Int. Conf. Mach. Learn., 2016, pp. 1842–1850.

[27] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in Proc. Int. Conf. Mach. Learn. Deep Learn. Workshop, 2015. [Online]. Available: https://www.cs.cmu.edu/~rshalicky/papers/one-shot1.pdf.

[28] J. Snell, K. Swersky, R. Zemel, and R. Zemel, “Prototypical networks for few-shot learning,” in Proc. Conf. Neural Inf. Process. Syst., 2017, pp. 4077–4087.

[29] S. Trinh, “Lifelong learning algorithms,” in Learning to Learn, Berlin, Germany: Springer, 1998, pp. 181–209.

[30] R. Vilalta and Y. Drissi, “A perspective view and survey of meta-learning,” Artif. Intell., vol. 180, no. 2, pp. 77–95, 2002.

[31] L. Bertinetto, J. F. Henriques, P. H. S. Torr, and A. Vedaldi, “Meta-learning with differentiable closed-form solvers,” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–15.

[32] W. Chen, Y. Liu, Z. Kira, Y. F. Wang, and J. Huang, “A closer look at few-shot classification,” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–16.

[33] M. Ren et al., “Meta-learning for semi-supervised few-shot classification,” in Proc. Int. Conf. Learn. Representations, 2018, pp. 1–15.

[34] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li, “Novel dataset for fine-grained image categorization: Stanford dogs,” in Proc. Int. Conf. Comput. Vis. Pattern Recognit. Workshop, 2011. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b5e3beb791cc17cdaf131d5cca6ceb796226d832.

[35] D. Kang, H. Kwon, J. Min, and M. Cho, “Relational embedding for few-shot classification,” in Proc. Conf. Neural Inf. Process. Syst., 2020, pp. 21981–21993.

[36] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, “The caltech-ucsd birds-200–2011 dataset,” 2011.
Xuesong Yang is currently working toward the graduate degree with the Department of Computer Science and Technology, Nanjing University, China. His research interests include few-shot learning and self-supervised learning.

Chuanqi Dong received the MSc degree from the Department of Computer Science and Technology, Nanjing University, China, in 2021. His research interests include machine learning and computer vision, especially on few-shot learning.

Pinzhuo Tian received the PhD degree from the Department of Computer Science and Technology, Nanjing University, in 2021. He is currently a lecturer with the School of Computer Engineering and Science of Shanghai University. His research interests lie in machine learning, including meta-learning and transfer learning.

Tiexin Qin received the BS degree in electronic information science and technology from the China University of Mining and Technology, China, in 2018, and the MS degree in computer science from Nanjing University, China, in 2021. He is currently working toward the PhD degree with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong. His research interests include machine learning and transfer learning.

Wenbin Li received the PhD degree from the Department of Computer Science and Technology, Nanjing University, in 2019. He is currently an associate researcher with the Department of Computer Science and Technology at Nanjing University, China. His research interests include machine learning and computer vision, particularly in metric learning, few-shot learning and their applications to image classification and image generation.

Ziyi Wang is currently working toward the graduate degree with the Department of Computer Science and Technology, Nanjing University, China. His research interests include few-shot learning and diffusion models.

Jing Huo received the PhD degree from the Department of Computer Science and Technology, Nanjing University, China, in 2017. She is currently an associate professor with the Department of Computer Science and Technology, Nanjing University. Her current research interests include machine learning and computer vision, with a focus on subspace learning, adversarial learning and their applications to heterogeneous face recognition and cross-modal face generation.
Yinghuan Shi is currently Associate Professor with the Department of Computer Science and Technology, Nanjing University, China. His research interests include machine learning, computer vision and medical image analysis. He has published more than 40 papers in CCF-A conference and IEEE Transactions. He was awarded as WuWenJun AI Excellent Young Scientist, Young Elite Scientists Sponsorship Program by CAST, and ACM Nanjing Rising Star.

Lei Wang (Senior Member, IEEE) received the PhD degree from Nanyang Technological University, Singapore. He is now professor with the School of Computing and Information Technology of University of Wollongong, Australia. His research interests include machine learning, pattern recognition, and computer vision. He has published more than 190 peer-reviewed papers, including those in highly regarded journals and conferences such as IEEE Transactions on Pattern Analysis and Machine Intelligence, International Journal of Computer Vision, CVPR, ICCV and ECCV, etc. He was awarded Early Career Researcher Award by Australian Academy of Science and Australian Research Council. He served as general co-chair of DICTA 2014 and area chair of ICIP 2019, and will serve as Program co-chair of ACCV 2022 in Macau.

Yang Gao received the PhD degree from the Department of Computer Science and Technology, Nanjing University, China, in 2000. He is currently a professor and also the deputy director of the Department of Computer Science and Technology, Nanjing University, where he is also directing the Reasoning and Learning Research Group. He has published more than 100 papers in top-tier conferences and journals. His current research interests include artificial intelligence and machine learning. He also serves as the program chair and area chair for many international conferences.

Jiebo Luo (Fellow, IEEE) is professor of computer science with the University of Rochester which he joined in 2011 after a prolific career of fifteen years with Kodak Research Laboratories. He has authored more than 500 technical papers and holds more than 90 U.S. patents. His research interests include computer vision, NLP, machine learning, data mining, computational social science, and digital health. He has been involved in numerous technical conferences, including serving as program co-chair of ACM Multimedia 2010, IEEE CVPR 2012, ACM ICMR 2016, and IEEE ICIP 2017, as well as general co-chair of ACM Multimedia 2018 and IEEE ICME 2024. He has served on the editorial boards of IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), IEEE Transactions on Multimedia (TMM), IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), IEEE Transactions on Big Data (TBD), ACM Transactions on Intelligent Systems and Technology (TIST), Pattern Recognition, Knowledge and Information Systems (KAIS), Machine Vision and Applications, and Intelligent Medicine. He is the current editor-in-chief of the IEEE Transactions on Multimedia. He is also a fellow of ACM, AAAI, SPIE, and IAPR.