Meta-Prototypical Learning for Domain-Agnostic Few-Shot Recognition

Rui-Qi Wang, Xu-Yao Zhang, *Senior Member, IEEE*, and Cheng-Lin Liu, *Fellow, IEEE*

***Abstract*— Few-shot learning (FSL) aims to classify novel images based on a few labeled samples with the help of meta-knowledge. Most previous works address this problem based on the hypothesis that the training set and testing set are from the same domain, which is not realistic for some real-world applications. Thus, we extend FSL to domain-agnostic few-shot recognition, where the domain of the testing task is unknown. In domain-agnostic few-shot recognition, the model is optimized on data from one domain and evaluated on tasks from different domains. Previous methods for FSL mostly focus on learning general features or adapting to few-shot tasks effectively. They suffer from inappropriate features or complex adaptation in domain-agnostic few-shot recognition. In this brief, we propose meta-prototypical learning to address this problem. In particular, a meta-encoder is optimized to learn the general features. Different from the traditional prototypical learning, the meta encoder can effectively adapt to few-shot tasks from different domains by the traces of the few labeled examples. Experiments on many datasets demonstrate that meta-prototypical learning performs competitively on traditional few-shot tasks, and on few-shot tasks from different domains, meta-prototypical learning outperforms related methods.**

***Index Terms*— Domain-agnostic few-shot recognition, image classification, meta-learning, prototypical learning.**

I. Introduction

Deep learning has reported dramatic performance improvements on many visual recognition tasks [1]–[6]. However, the ability to learn novel concepts with only a few labeled examples is still the hallmark of real intelligence such as humans. Aiming to fill in the gap between machine intelligence and human intelligence, few-shot learning (FSL) draws wide attention in the research community [7]–[13].

FSL is a problem that requires a model to distinguish images of novel classes with only a few labeled examples. Unlike common classification that requires sufficient training data for target classes, in FSL, rich labeled samples from completely different classes are available. These samples are used to learn the meta knowledge to mimic the prior knowledge of humans. For example, these data can be used to learn a feature encoder that embeds visual attributes. Generally, there is no class overlap among the training, testing, and validation sets in FSL. The evaluation protocol is as follows: Many few-shot classification tasks consisting of the support set of shots (a few labeled examples) and the query set of queries (data to be recognized) are sampled from the testing set. For each few-shot task, the model is required to classify queries based on the given shots. If the support set consists of k shots for each of n classes, the task is called n-way k-shot. The performance is measured by the mean accuracy with a confidence interval of many sampled tasks. Recent methods usually use a meta-learning strategy to mimic the testing protocol during training. Technically, the training set is also sampled into many few-shot tasks and the model is learned according to the classification loss of these tasks. Previous methods on FSL can be divided into two classes: Some methods focus on learning a feature encoder as the meta-knowledge that can be directly applied to novel classes [10], [13], [14], and the support set is usually only used to generate classification weights such as prototypes. Other methods focus on adapting the meta knowledge to each few-shot task with the help of the support set [15], [16].

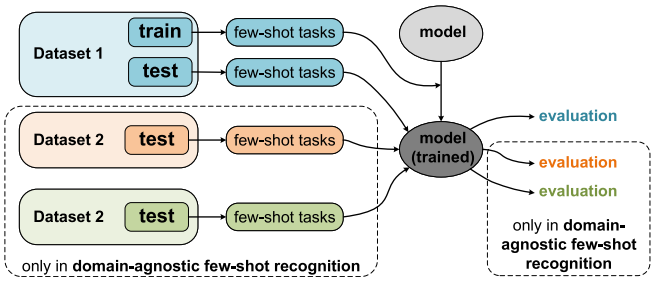

Fig. 1. Illustration of domain-agnostic few-shot recognition in comparison with conventional FSL. Domain-agnostic few-shot recognition focuses more on few-shot tasks from different domains (datasets).

Most previous methods neglect the fact that the testing tasks can come from different domains in real-world applications. We name this problem domain-agnostic few-shot recognition as shown in Fig. 1, where the model is evaluated by testing few-shot tasks from different domains. Existing methods for FSL suffer from the agnostic domains. For the methods focusing on general features, the encoder lacks adaptation ability and the performance is limited. For the methods focusing on effective adaptation, the learned meta-knowledge can adapt to tasks of new domains, but the procedure is complex and the final performance is severely limited. Thus, we propose meta-prototypical learning to address the domain-agnostic few-shot recognition problem. Technically, meta-prototypical learning incorporates prototypical learning [14] and model-agnostic meta-learning (MAML) [16] to learn a meta feature encoder that can adapt to novel tasks efficiently by a few steps of adaptation based on shots.

Our contributions can be summarized as follows: First, we extend FSL to domain-agnostic few-shot recognition, in which the domain of the testing task is changed. Second, we formulate meta-prototypical learning for domain-agnostic few-shot recognition, and an effective first-order approximation is introduced to train the meta encoder. Experiments on different datasets show that the meta-encoder can adapt to new tasks effectively and achieve better performance in domain-agnostic few-shot recognition problems.

Manuscript received September 22, 2019; revised August 8, 2020 and February 19, 2021; accepted May 18, 2021. This work was supported in part by the National Key Research and Development Program under Grant 2018AAA0100400, in part by the National Natural Science Foundation of China (NSFC) under Grant 62076236 and Grant 61721004, in part by the Key Research Program of Frontier Sciences of CAS under Grant ZDBS-LY-7004, and in part by the Youth Innovation Promotion Association of CAS under Grant 2019141. *(Corresponding author: Cheng-Lin Liu.)*

Rui-Qi Wang and Xu-Yao Zhang are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: ruiqi.wang@nlpr.ia.ac.cn; xyz@nlpr.ia.ac.cn).

Cheng-Lin Liu is with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing 100190, China (e-mail: liucl@nlpr.ia.ac.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3083650.

Digital Object Identifier 10.1109/TNNLS.2021.3083650

The rest of this brief is organized as follows. Section II describes related works. Section III introduces the proposed meta-prototypical learning method. Section IV presents experimental results. At last, Section V draws concluding remarks.

II. Related Works

A. Few-Shot Learning

FSL aims to gain classification ability based on only a few examples with meta-knowledge. Some methods focus on learning general representation and classify queries by matching them with shots or prototypes of shots. Vinyals *et al.* [10] proposed a matching network to recognize unlabeled data (queries) based on labeled data (shots) using matching measurement between their attentive embeddings. Furthermore, Snell *et al.* [14] proposed a prototypical network to classify queries based on prototypes derived from shots. Meta-learning is another widely used strategy for FSL. Ravi and Larochelle [15] proposed an LSTM [17] based meta-learner to train a custom model for each few-shot task. Gidaris and Komodakis [18] utilized a meta-learning generated classifier to classify novel few-shot classes. Besides, there are also image retrieval-based FSL frameworks [19] and reinforcement learning based ones [20].

The proposed meta-prototypical learning is closely related to prototypical network [14] and MAML [16]. We take advantage of the general feature encoder in [14] and the efficient adaptation of [16] to learn a meta encoder that adapts to few-shot tasks from different domains efficiently and effectively.

B. Meta-Learning

Meta-learning [21]–[23] enables a model to learn fast on new missions and is also called "learning to learn." Some researchers addressed this problem using a meta-learner that learns how to optimize the parameters of the original model [24]–[26]. These approaches can also be applied to deep networks [27]–[29]. Recent approaches also learn weight initialization for few-shot image recognition [15], [16]. Another type of approach for meta-learning takes advantage of recurrent networks, where a recurrent learner will adapt to new tasks by unfolding the recurrent layers [11], [12].

Among all these methods, we follow MAML [16] that has good scalability and can be easily integrated with other methods. Generally, for a given model, MAML learns a meta-initialization that adapts to new tasks effectively based on gradient. Different from traditional learning methods, the meta-model is optimized based on losses after a few steps of adaptation on sampled tasks. This indicates that MAML focuses on performance after several updates. As a result, it learns an initialization that can adapt to different tasks effectively. We incorporate MAML into prototypical learning to learn the meta encoder that can adapt to new domains effectively.

C. Prototypical Learning

Prototypical learning is a classical method in pattern recognition. The earliest method for prototypical learning is a k-nearest neighbor (k-NN). Then learning vector quantization (LVQ) [30] proposed using learnable prototypes for classification. Traditional prototypical learning methods are mainly based on manually designed features. Recently, the combination of prototypical learning and deep learning draws increasing attention and demonstrates superiority in robust classification [31] and few-shot recognition [14]. With deep neural networks, prototypical learning can now use convolutional neural networks (CNNs) as the feature extractor other than manually designed features.

Meta-prototypical learning incorporates prototypical learning to optimize the encoder, with which the classification weights can be calculated based on shots. Thus, the adaptation process can be shortened by a relatively good performance other than a random guess at the beginning.

III. Meta-Prototypical Learning

Consider the prototypical learning method in FSL, the optimization of the model can be formulated as follows:

$$\theta = \arg\min_{\theta} \ \mathbb{E}_{\mathcal{T}}\{L(\mathcal{T};\theta)\} \tag{1}$$

where θ is the optimizable parameter of the model (usually feature encoder), L is the classification loss function and $\mathcal{T}$ is a few-shot task sampled from the training set. $\mathcal{T}$ contains support set $\mathcal{S}$ and query set $\mathcal{Q}$. Labeled samples in $\mathcal{S}$ are used to generate class prototypes to classify samples in $\mathcal{Q}$, based on which the loss is constructed.

Prototypical learning considers learning a general encoder without adaptation on specific few-shot tasks $\mathcal{T}$. However, in meta-prototypical learning, we optimize θ by making it adapt effectively, which can be formulated as follows:

$$\theta = \arg\min_{\theta} \ \mathbb{E}_{\mathcal{T}}\{L_{\text{meta}}(\mathcal{T};\phi = \arg\min_{\phi} L_S(\mathcal{S},\phi;\phi_0 = \theta))\} \tag{2}$$

where L_{meta} is the classification loss of a few-shot task $\mathcal{T}$ and L_S is the loss for adapting feature encoder parameter ϕ based on gradients from support set $\mathcal{S}$ of $\mathcal{T}$. And θ is the initialization of ϕ for all few-shot tasks. Different from prototypical learning, meta-prototypical learning optimizes θ by solving a bilevel optimization problem as shown in (2). The inner problem is adapting the parameter of feature encoder ϕ for each specific few-shot task. And the outer problem is optimizing θ, the parameter of the meta-encoder (meta-initialization) for all few-shot tasks.

Fig. 2 illustrates our meta-prototypical learning process. Given a meta-encoder $f(\cdot;\theta)$, the final goal is to find the meta-initialization θ that can adapt to different tasks effectively with gradients of support set $\mathcal{S}$, which is called inner update. After several steps of inner update, a loss can be constructed based on the classification results of samples in the query set $\mathcal{Q}$ to evaluate the adaptation capability of θ and calculate gradients. The optimization process of the parameter of meta-encoder θ is called meta update.

The motivation of meta-prototypical learning is to consider adaptation during training. Prototypical learning demonstrates strong generalization and robustness in previous work [31]. But in domain-agnostic few-shot recognition, the learned feature may fail because of changed domains. Thus, adaptation to the domain of the testing task is required. We incorporate MAML [16] to achieve a meta-learned prototype encoder that can adapt to tasks of changed domains within a few inner updates on the support set. It should be noticed that different from MAML, inner update and meta update in our meta-prototypical learning framework are based on different losses. For inner update, we directly optimize the distances between embeddings in the support set. For meta update, we use the prototypical classification loss on the query set.

A. Inner Update

Inner update adapts the encoder to few-shot recognition tasks from different domains. It is performed based on the support set according to a supervised clustering loss that drives the embeddings of shots in support set $\mathcal{S}$ to be compact within classes and separate between classes. With given distance metric $\mathcal{M}(\cdot,\cdot)$, the inner update loss L_s

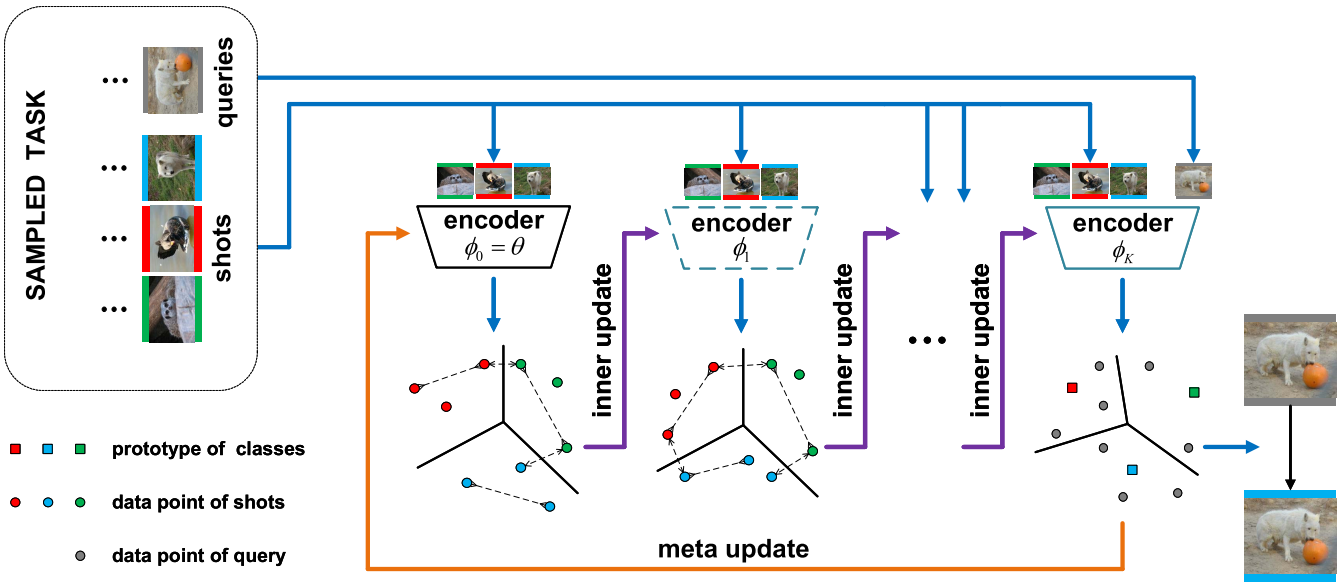

Fig. 2. Training of meta-prototypical learning. For a specific few-shot task, the meta encoder is used as the initialization of the encoder. It is updated by minimizing the largest sample distances within classes and maximizing the shortest sample distances between classes in support set $\mathcal{S}$ (inner update). Finally, samples in the query set $\mathcal{Q}$ are classified based on prototypes generated with shots. And the parameter of the meta encoder is updated based on classification loss of queries (meta update).

can be formulated as follows:

$$L_s(\mathcal{S};\phi) = \sum_{i,j} \mathcal{M}(f(x_i;\phi), f(x_j;\phi)) - \sum_{m,n} \mathcal{M}(f(x_m;\phi), f(x_n;\phi)) \quad (3)$$

where $x_i, x_j \in \mathcal{S}$ represent two images from the same class and $x_m, x_n \in \mathcal{S}$ are two images from different classes. And $f(\cdot;\phi)$ is the encoder for this specific few-shot task with learnable parameter ϕ. In the implementation, we choose the largest embedding distances within the classes and the smallest embedding distances between classes to fulfill the inner update. The optimization of inner update based on gradient descent can be written as

$$\phi_{k+1} = \phi_k - \alpha \nabla_{\phi_k} L_{s_k}, \quad k = 0, 1, 2, \ldots \quad (4)$$

where ϕ_k indicates parameter after kth inner update and $\phi_0 = \theta$ is initialized with the meta-initialization. L_{s_k} is the loss for inner update at kth step.

B. Meta Update

Meta update solves the outer problem of (2) to optimize the meta-encoder parameter (meta-initialization) θ, which depends on classification loss on the query set $\mathcal{Q}$ using the encoder parameter after the last inner update ϕ_K. After inner updates, the encoder maps images from support and query set into the same embedding space. The embeddings of shots are averaged as class prototypes $\mathcal{P}_c$. With the distance metric $\mathcal{M}(\cdot,\cdot)$, the distance between a query image q_j and the prototype of cth class is computed in embedding space as $\text{dist}_{jc} = \mathcal{M}(f(q_j;\phi_K), \mathcal{P}_c)$. Then the probability p_{jc} that the jth query image is from class c is computed based on these distances as

$$p_{jc} = \frac{\exp(-\text{dist}_{jc})}{\sum_{k=1}^{C} \exp(-\text{dist}_{ik})} \quad (5)$$

where C is the number of classes. Meta loss is constructed simply by maximizing the probability that all images are classified correctly. It can be formulated as

$$L_{\text{meta}} = -\frac{1}{N} \sum_{j=1}^{N} \log p_{jz_j} \quad (6)$$

where p_{jz_j} is the probability that a query image $q_j \in \mathcal{Q}$ belongs to its genuine class z_j. N is the total number of queries in a few-shot task. It should be noticed that L_{meta} is constructed based on ϕ_K and used to optimize θ (initialization ϕ_0 at the beginning of inner update), which makes the optimization complicated. Fortunately, the relation between ϕ_K and θ is simple as (4) shows. Based on this, we can formulate the gradient for meta update with inner update learning rate α as follows:

$$\begin{aligned}\frac{\partial L_{\text{meta}}}{\partial \theta} &= \frac{\partial L_{\text{meta}}}{\partial \phi_0} = \frac{\partial L_{\text{meta}}}{\partial \phi_K} \cdot \frac{\partial \phi_K}{\partial \phi_{K-1}} \cdots \frac{\partial \phi_1}{\partial \phi_0} \\ &= \frac{\partial L_{\text{meta}}}{\partial \phi_K} \cdot \left(1 - \alpha \nabla^2_{\phi_{k-1}} L_{s_{k-1}}\right) \cdots \left(1 - \alpha \nabla^2_{\phi_0} L_{s_0}\right) \\ &= \frac{\partial L_{\text{meta}}}{\partial \phi_K} \cdot \left(1 - \alpha \sum_{i=0}^{k-1} \nabla^2_{\phi_i} L_{s_i} + \alpha^2 \sum_{i=0}^{k-1} \sum_{j=0, j\neq i}^{k-1} \nabla^2_{\theta_i} L_{s_i} \right. \\ &\qquad \left. \cdot \nabla^2_{\phi_j} L_{s_j} - \alpha^3 \sum \sum \sum + \cdots \right). \end{aligned} \quad (7)$$

$(\partial L_{\text{meta}}/\partial\theta)$ will be multiplied by a small meta-learning rate β in meta update. Thus, all terms with second-order derivatives have little effect on meta update and can be omitted because they are further multiplied by α. This is a first-order approximation, which is also mentioned and evaluated to be effective in [16]. Readers can refer to [32] for further analysis. Meta update with first-order approximation can be formulated as follows:

$$\theta \leftarrow \theta - \beta \cdot \frac{\partial L_{\text{meta}}}{\partial \phi_K} \quad (8)$$

where $\theta = \phi_0$ is the meta-initialization. ϕ_K is the parameter after K steps of inner update. The first-order approximation not only simplifies the computation but also reduces the requirement of memory, because we no longer need to store all K inner update

Fig. 3. Samples of different datasets. The domain gaps between mini-ImageNet and others are visually large.

processes except the last one. Generally, we can save K times memory space with first-order approximation in the inner update process for meta-prototypical learning with K steps of inner update.

IV. Experiments

In this section, we evaluate the proposed meta-prototypical learning method in comparison with prototypical network [14] and MAML [16]. All methods were optimized on the training set of mini-ImageNet [15] and evaluated on test tasks from different datasets including mini-ImageNet, MNIST [33], CIFAR-10 [34], and Fashion-MNIST [35]. All experiments were implemented with PyTorch [36] 0.4.0. Code will be released at https://github.com/RuiqiWang95/meta-prototypical-learning.

A. Datasets, Protocols, and Implementation Details

1) Datasets: Four datasets were used in our experiments: mini-ImageNet, MNIST, CIFAR-10, and Fashion-MNIST. The dataset of mini-ImageNet [15] consists of 100 classes with 600 images selected from ImageNet [37] for each class. The 100 classes are split into a training set (64 classes), validation set (16 classes), and test set (20 classes). MNIST [33] is a monochrome handwritten digit dataset of ten classes with 60 K images for training and 10 K images for testing. Similarly, fashion-MNIST [35] is a monochrome fashion dataset with ten classes. CIFAR-10 [34] includes ten classes of color images, which has 50 K images for training and 10 K images for testing. The testing sets of these four datasets were used to sample few-shot tasks from different domains for evaluation, while the models were trained only on the training set of mini-ImageNet. Samples from the four datasets are visualized in Fig. 3 to demonstrate the strong domain gaps.

2) Protocols: We review the training and evaluation protocols used in previous works [14], [16] and introduce the adjustments for domain-agnostic evaluation. For both conventional and domain-agnostic few-shot recognition, the training set of mini-ImageNet is sampled into many few-shot tasks for optimization. The trained model is evaluated with few-shot tasks sampled from the testing set. A few-shot task consists of a support set (few-shot training set) and a query set. Samples in the query set are classified based on the support set. The performance is measured as mean accuracy with a confidence interval of many (e.g., 1000) few shot tasks of the same setting (e.g., 5-way 5-shot). For domain-agnostic few-shot recognition, the model is evaluated with not only the testing set of mini-ImageNet but also testing sets from other datasets of different domains.

3) Implementation Details: All images shared the same preprocessing: resized to 84 $\times$ 84 pixels and normalized. Euclidean distance was used as the metric $\mathcal{M}$ in the experiments as in [14]. The encoder (feature extractor) remained the same as that of prototypical network [14] and MAML [16], which consisted of four convolutional blocks. Each block contained a 3 $\times$ 3 convolutional layer of 64 filters, a batch normalization [38], a ReLU nonlinear activation, and a 2 $\times$ 2 max-pooling layer. Meta-prototypical learning and prototypical learning [14] compute prototypes of classes for classification. Differently, MAML contains an additional linear layer as the classifier. For meta-prototypical learning and prototypical learning, the training set was sampled into many 20-way 5-shot tasks with 15 queries for each class. For MAML, it was trained with sampled tasks of n-way 5-shot 15 queries, and n was decided by the target evaluation task. All models were optimized with Kinga and Adam [39] optimizer with a learning rate $\beta = 1e-3$ if not specified. The learning rate was divided by 2 after every 2000 tasks during the whole training process of 60 000 sampled tasks. For MAML and meta-prototypical learning, the default inner-update learning rate α_{tr} were $1e-2$ and $1e-4$ by default, and three steps of adaptation were performed during training. In evaluation, 1000 tasks were sampled from the testing set and the mean accuracy with a confidence interval with 95% confidence was reported. Evaluation tasks for mini-ImageNet were 5-way 5-shot tasks, to be consistent with previous methods. And those from datasets of different domains were 10-way 5-shot tasks.

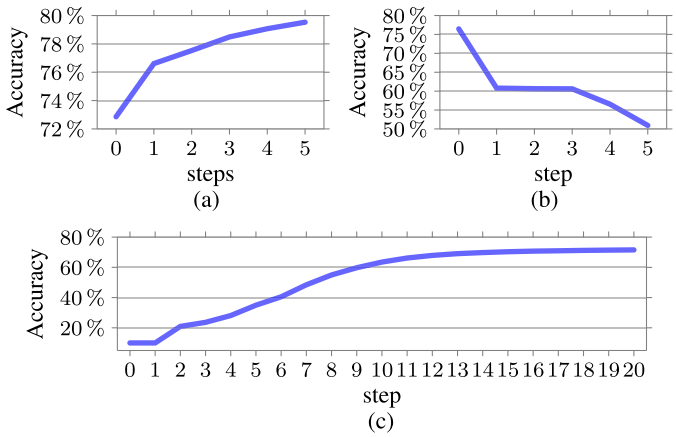

Fig. 4. Typical adaptation process of the three methods on MNIST while trained on mini-ImageNet. The horizontal axis demotes steps of inner update and the vertical axis denotes performance. (a) Meta-prototypical learning. (b) Prototypical network. (c) MAML.

TABLE I

Experimental Results on Mini-ImageNet Few-Shot Recognition With 5-Way 5-Shot Condition. Results are Provided as Averaged Accuracy and Confidence Interval With 95% Confidence Over 600 Tasks. "*" Means Our Implementation. With Our Implementation, Five Steps of Inner Update are Performed in MAML and Meta-Prototypical Learning

Model	Accuracy	Confidence Interval
fine-tuning baseline [11]	49.79%	±0.79%
nearest neighbor baseline [11]	51.04%	±0.65%
matching net [10]	55.31%	±0.73%
meta-learn LSTM [15]	60.60%	±0.71%
MAML [11]	63.11%	±0.92%
prototypical net [14]	68.20%	±0.66%
relations net [13]	65.32%	±0.70%
prototypical net*	67.32%	±0.51%
MAML*	62.29%	±0.53%
meta-prototypical learning (Ours)	67.19%	±0.50%

B. Domain-Agnostic Few-Shot Recognition Experiments

Meta-prototypical learning is proposed to address few-shot recognition tasks with different domains that are agnostic during training. Evaluation on mini-ImageNet (the same domain as the training set)

TABLE II

EXPERIMENTAL RESULTS ON AGNOSTIC TASKS FROM MNIST, CIFAR-10, AND FASHION-MNIST. "@k" INDICATES THE PERFORMANCE AFTER kTH STEP OF INNER UPDATE. ALL THREE DATASETS ARE SAMPLED INTO 10-WAY 5-SHOT TASKS. MODELS USED HERE ARE TRAINED ON MINI-IMAGENET. PERFORMANCES ARE PROVIDED AS AVERAGED ACCURACY AND CONFIDENCE INTERVAL WITH 95% CONFIDENCE OVER 1000 TASKS. T SHOWS THE MULTIPLE RELATIONSHIPS BETWEEN DIFFERENT METHODS AND PROTOTYPICAL NETWORK [14]

Method	*Running time (T)*	*MNIST*	*CIFAR-10*	*Fashion-MNIST*
prototypical nets [14]	1×	76.20±0.26%	30.87±0.25%	66.08±0.27%
MAML [16] @5	28×	34.97±0.61%	23.01±0.23%	38.48±0.39%
MAML [16] @10	34×	59.30±0.49%	26.47±0.24%	53.54±0.33%
MAML [16] @20	40×	71.54±0.33%	26.06±0.25%	57.73±0.29%
meta-prototypical learning (Ours)@5	35×	**79.53±0.24%**	**32.55±0.27%**	**67.52±0.25%**

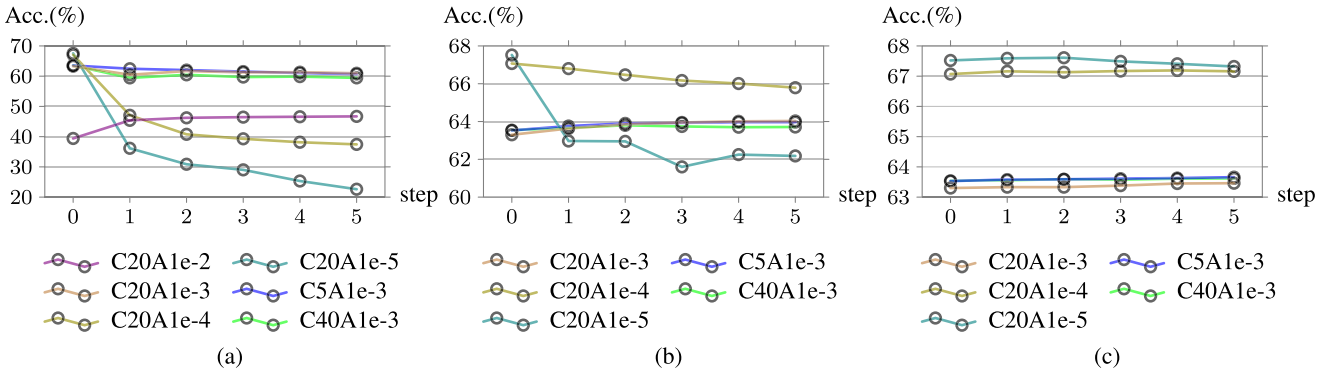

Fig. 5. Hyper-parameter analysis on mini-ImageNet. In "$CxAy$," x is the class number of training few-shot tasks and y is the α_{tr} in training. The horizontal axis denotes steps of inner update and the vertical axis denotes the performance. (a) $\alpha_{\text{te}} = 1e-3$. (b) $\alpha_{\text{te}} = 1e-4$. (c) $\alpha_{\text{te}} = 1e-5$.

is provided in Table I with comparison to previous methods. The inner update learning rate was $1e-5$ for meta-prototypical learning because these evaluation tasks were from the same domain as that of training tasks. Table II demonstrates results on three datasets of different domains, in which the inner update learning rate was $1e-3$.

1) Conventional Few-Shot Recognition: Table I shows that the proposed meta-prototypical learning method reports competitive performance compared with previous methods on conventional few-shot recognition evaluation. That is, when evaluated with tasks of the same domain as training tasks, meta-prototypical learning performs closely to prototypical learning and outperforms MAML by 5%.

2) Domain-Agnostic Few-Shot Recognition: In Table II, we compare meta-prototypical learning, prototypical learning, and MAML on tasks from different domains. Meta-prototypical learning reported the best performance on three different datasets. By comparing MAML with different adaptation steps, we can see that it is still effective, but the performance is limited and the adaptation is time-consuming. The difference between meta-prototypical learning and prototypical learning lies in that meta-prototypical learning can adapt effectively. By comparing the two methods on three datasets, it is clear that the adaptation helps meta-prototypical learning to improve performance on tasks from different domains. Further, meta-prototypical learning with 5 adaptation steps outperforms MAML with 20 adaptation steps. This is a result of the different strategies to compute classification weights. MAML adapts to few-shot tasks with a classifier that performs random guesses at the beginning. In comparison, meta-prototypical learning computes the prototypes as the classification weights, which provides decent performance at the beginning and shortens the adaptation. This is more clearly demonstrated in Fig. 4 and Table III. Thus, we can draw the conclusion that meta-prototypical learning is more effective than MAML on tasks from different domains. However, meta-prototypical learning is 35 times slower than prototypical network, because prototypical network needs no adaptation. And this is the main limitation of our proposed method, which requires further investigations in the future works.

3) Adaptation Process: Although prototypical learning considers no adaptation during training, it can also be forced to "adapt" to few-shot tasks using the same strategy in meta-prototypical learning. We visualize the adaptation processes of meta-prototypical learning, prototypical network, and MAML in Fig. 4. The inner update learning rates for adaptation α_{te} were $1e-2$ for MAML and $1e-3$ for both prototypical learning and meta-prototypical learning. The experiments show that MAML can adapt to specific tasks, but the process is slow and time-consuming. The encoder trained with prototypical learning fails to adapt. In comparison, meta-prototypical learning adapts to tasks effectively and achieves the best performance. These validate that meta-prototypical learning has a stronger adaptation ability than prototypical learning because the learning of meta encoder considers adaptation in optimization. And by learning the encoder and generating classification weights based on the support set, meta-prototypical learning has a much higher initial performance than MAML, which shortens the adaptation process significantly and provides better final performance.

C. Hyper-Parameter Analysis

1) Hyper-Parameters of Meta-Prototypical Learning: Meta-prototypical learning involves hyper-parameters that may influence the performances. We focused on the class number of each task during training and the inner update learning rate in training and testing (α_{tr} and α_{te}). Because in meta-prototypical learning, n-way of training tasks can be different from that of testing tasks. And α controls inner update for adaptation, which is important in meta-prototypical learning. We evaluated models trained with different hyper-parameters on mini-ImageNet (the same domain) and MNIST (different domain).

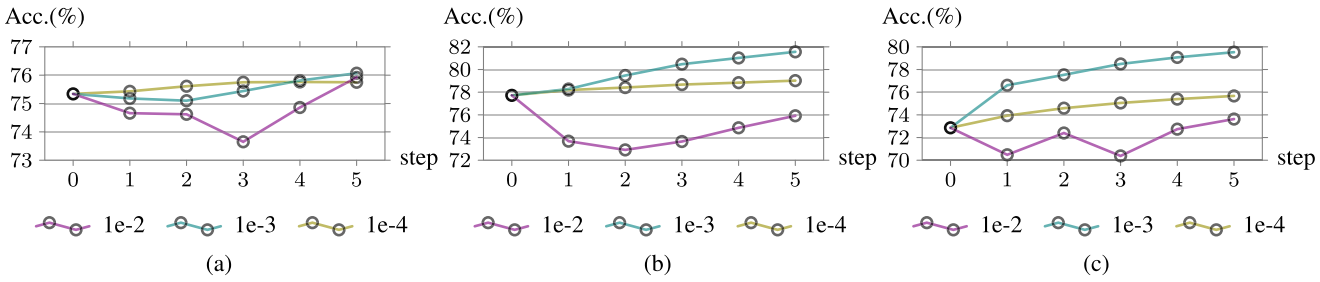

Fig. 6. Hyper-parameter analysis on MNIST. Results of models trained with the same inner update learning rate α_{tr} are organized in the same line chart. Different lines indicate different inner update learning rates α_{te}. The horizontal axis denotes steps of inner update and the vertical axis denotes the performance. (a) $\alpha_{\text{tr}} = 1e-2$. (b) $\alpha_{\text{tr}} = 1e-3$. (c) $\alpha_{\text{tr}} = 1e-4$.

TABLE III

HYPER-PARAMETER COMPARISONS OF PROTOTYPICAL LEARNING [14], MAML [16], AND META-PROTOTYPICAL LEARNING (OURS). WE FOCUS ON THE INNER-UPDATE LEARNING RATE DURING TRAINING α_{tr} AND TESTING α_{te}. "@k" INDICATES THE PERFORMANCE AFTER THE kTH STEP OF INNER UPDATE. "*" NOTES THE HYPER-PARAMETER CHOICE THAT FAILS

Method	*Hyper-parameter*		*MNIST*		*CIFAR-10*		*Fashion-MNIST*	
	α_{tr}	α_{te}	@0	@5	@0	@5	@0	@5
prototypical networks	-	1e-3	76.20±0.26%	50.93±0.28%	30.87±0.25%	11.86±0.23%	66.08±0.27%	22.38±1.25%
		1e-4	76.20±0.26%	74.02±0.28%	30.87±0.25%	26.60±0.27%	66.08±0.27%	63.47±0.27%
MAML	1e-2	1e-2	10.00±0.01%	34.97±0.61%	10.04±0.02%	23.01±0.23%	9.99±0.03%	38.48±0.39%
		1e-3	10.00±0.01%	16.35±0.24%	10.04±0.02%	11.28±0.17%	9.99±0.03%	15.56±0.24%
	1e-3 *	1e-2	9.94±0.12%	9.93±0.12%	10.00±0.18%	10.31±0.18%	9.89±0.25%	10.41±0.26%
		1e-3	9.94±0.12%	9.89±0.12%	10.00±0.18%	10.03±0.18%	9.89±0.25%	9.94±0.25%
meta-prototypical learning	1e-3	1e-3	77.72±0.21%	81.56±0.24%	30.68±0.26%	31.72±0.26%	62.08±0.27%	65.84±0.25%
		1e-4	77.72±0.21%	79.02±0.24%	30.68±0.26%	32.31±0.26%	62.08±0.27%	64.48±0.27%
	1e-4	1e-3	72.86±0.21%	79.53±0.25%	31.22±0.26%	32.55±0.27%	64.73±0.26%	67.52±0.25%
		1e-4	72.86±0.21%	75.67±0.24%	31.22±0.26%	32.15±0.27%	64.73±0.26%	65.31±0.26%

Fig. 5 shows the experimental results of different hyper-parameter settings on mini-ImageNet. As the results demonstrated, the adaptation process on mini-ImageNet hardly benefited the performance, because the testing tasks were from the same domain as that of training data. And larger inner update learning introduced over-fitting to the few support samples, which resulted in lower performance. By comparing the results of different training ways (class number of training tasks), we find that it has little influence on the performance.

Fig. 6 shows the results of different α_{tr} and α_{te} on MNIST, in which the adaptation process is clearly beneficial for tasks from different domains. We can see that among all settings, $\alpha_{\text{tr}} = \alpha_{\text{te}} = 1e-3$ achieves the best performance of 81.56%. And model trained with $\alpha_{\text{te}} = 1e-2$ reported an unstable adaptation process.

2) Hyper-Parameter of the Three Methods: We also report results of different hyper-parameter choices of the three methods in Table III. As the experiments show, prototypical network [14] fails to adapt with different hyper-parameter choices. MAML [16] suffers from limited performance and is sensitive to α_{tr}. MAML trained with $\alpha_{\text{tr}} = 1e-3$ failed to converge and the model showed no effect. Because MAML adapts to different tasks from random guess, which makes smaller α_{tr} hard to show effect and fails to optimize the meta-model. In comparison, the proposed meta-prototypical learning can adapt effectively and robustly. If we choose different hyper-parameters for different target datasets, the performance can be further boosted, such as 79.53% → 81.56% on MNIST.

V. CONCLUSION

In this brief, we investigate the domain-agnostic few-shot recognition, where the testing tasks have different domains from training tasks. To address this problem, we propose meta-prototypical learning that adapts a meta encoder to different few-shot recognition tasks. Technically, meta-prototypical learning takes advantage of prototypical learning to get a good initial performance and meta-learning to achieve efficient adaptation. The adaptation capability of the encoder is successfully improved by adding the adaptation as the inner problem into the optimization of the encoder. Experimental results on various datasets demonstrate that adapting the meta-encoder to specific tasks helps to improve the performance for domain-agnostic few-shot recognition. Moreover, in implementation, meta-prototypical learning is more robust to hyper-parameters compared with MAML. The main limitation of meta-prototypical learning is computational efficiency. In the future, we will try to find more advanced methods with higher efficiency to address domain-agnostic few-shot recognition.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[4] P. Tang, X. Wang, B. Shi, X. Bai, W. Liu, and Z. Tu, "Deep FisherNet for image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 7, pp. 2244–2250, Jul. 2019.

[5] H. Zhu, L. Jiao, W. Ma, F. Liu, and W. Zhao, "A novel neural network for remote sensing image matching," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2853–2865, Sep. 2019.

[6] W. Luo, J. Li, J. Yang, W. Xu, and J. Zhang, "Convolutional sparse autoencoders for image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3289–3294, Jul. 2018.

[7] M. G. Miller, N. E. Matsakis, and P. A. Viola, "Learning from one example through shared densities on transforms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2000, pp. 464–471.

[8] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts," in *Proc. Annu. Meeting Cogn. Sci. Soc.*, 2011, pp. 1–7.

[9] 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://sites.google.com/site/deeplearning2015/37.pdf

[10] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3637–3645.

[11] 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.

[12] T. Munkhdalai and H. Yu, "Meta networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2554–2563.

[13] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1199–1208.

[14] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4077–4087.

[15] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. Int. Conf. Learn. Represent.*, 2017. [Online]. Available: https://openreview.net/pdf?id=rJY0-Kcll

[16] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic metalearning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4367–4375.

[19] E. Triantafillou, R. Zemel, and R. Urtasun, "Few-shot learning through an information retrieval lens," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2255–2265.

[20] A. Puzanov and K. Cohen, "Deep reinforcement one-shot learning for artificially intelligent classification systems," 2018, *arXiv:1808.01527*. [Online]. Available: http://arxiv.org/abs/1808.01527

[21] S. Thrun and L. Pratt, *Learning to Learn*. New York, NY, USA: Springer, 2012.

[22] J. Schmidhuber, "Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... Hook," Ph.D. dissertation, Institut f. Informatik, Technische Universität München, München, Germany, 1987.

[23] D. K. Naik and R. J. Mammone, "Meta-neural networks that learn by learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 1992, pp. 437–442.

[24] Y. Bengio, S. Bengio, and J. Cloutier, "Learning a synaptic learning rule," in *Proc. Seattle Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 2, Jul. 1991, p. 969.

[25] J. Schmidhuber, "Learning to control fast-weight memories: An alternative to dynamic recurrent networks," *Neural Comput.*, vol. 4, no. 1, pp. 131–139, Jan. 1992.

[26] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei, "On the optimization of a synaptic learning rule," in *Proc. Preprints Conf. Optimality Artif. Biol. Neural Netw.* Austin, TX, USA: Univ. of Texas, 1992, pp. 6–8.

[27] M. Andrychowicz *et al.*, "Learning to learn by gradient descent by gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3981–3989.

[28] H. Edwards and A. Storkey, "Towards a neural statistician," in *Proc. Int. Conf. Learn. Represent.*, 2017. [Online]. Available: https://openreview.net/pdf?id=HJDBUF5le

[29] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," in *Proc. Int. Conf. Learn. Represent.*, 2017. [Online]. Available: https://openreview.net/pdf?id=rkpACe1lx

[30] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, nos. 1–3, pp. 1–6, 1998.

[31] H.-M. Yang, X.-Y. Zhang, F. Yin, and C.-L. Liu, "Robust classification with convolutional prototype learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3474–3482.

[32] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018, *arXiv:1803.02999*. [Online]. Available: http://arxiv.org/abs/1803.02999

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[34] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 7, 2009.

[35] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MBIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*. [Online]. Available: https://arxiv.org/abs/1708.07747

[36] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035. [Online]. Available: https://github.com/pytorch/pytorch/blob/master/CITATION

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[39] J. Ba and D. Kingma, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, vol. 5, 2015. [Online]. Available: https://arxiv.org/abs/1412.6980