Pattern Recognition 129 (2022) 108704

Contents lists available at ScienceDirect

Pattern Recognition

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

A novel forget-update module for few-shot domain generalization

Minglei Yuan[a], Chunhao Cai[a], Tong Lu[a,*], Yirui Wu[b], Qian Xu[a], Shijie Zhou[c]

[a] National Key Lab for Novel Software Technology, Nanjing University, Nanjing, China
[b] College of Computer and Information, Hohai University, Nanjing, China
[c] Jiangsu Welm Technology Co. Ltd, Nantong, China

ARTICLE INFO

Article history:
Received 13 October 2020
Revised 31 March 2022
Accepted 7 April 2022
Available online 12 April 2022

Keywords:
Few-shot classification
Domain adaptation
Few-shot domain generalization

ABSTRACT

Existing Few-Shot Learning (**FSL**) methods learn and recognize new classes with the help of prior knowledge. However, they cannot handle this task well in a cross-domain scenario when training and testing sets are from different domains, since the fact that prior knowledge in different domains often varies greatly. To solve this problem, in this paper, we propose a few-shot domain generalization method, which is designed to extract relationship embeddings using Forget-Update Modules named **FUM**. The relationship embedding considers valuable relational information between samples in a specific task, and the forget-update module takes into account differences between domains and adjusts the distribution of relational embeddings through forgetting and updating mechanisms based on specific tasks. To evaluate the few-shot domain generalization ability of FUM, extensive experiments on eight cross-domain scenarios and six same-domain scenarios are conducted, and the results show that FUM achieves superior performances compared to recent few-shot learning methods. Visualization results also show that the distribution of the relationship embeddings extracted by FUM has stronger few-shot domain generalization ability than the feature embeddings used in the existing FSL methods.

1. Introduction

Many deep learning models [1–4] have achieved remarkable results on object recognition tasks. These powerful deep learning methods heavily rely on deep neural networks trained with thousands of label instances. However, a large scarcity of labeled instances requires huge manual work, which is a time-consuming and annoying task and limits working scenarios of deep learning methods. Most of these popular deep learning models will encounter the overfitting problem when the training data is limited. However, people can often successfully generalize new concepts from just a single example by action, imagination, and explanation [5]. The problem of learning to generalize to new classes with a limited number of label examples, called few-shot learning (**FSL**), has attracted considerable attention in the past few years. Recently, research on FSL begin to appear in many areas, such as image classification [6–10], gesture recognition [11], activity recognition [12], and logo retrieval [13].

Many FSL methods have been proposed to learn new transformable visual concepts with limited samples in recent years. Some FSL methods are often trained and evaluated on the same dataset, where the basic and novel classes are from the same domain (e.g., generic object recognition tasks). However, in real scenarios, training and testing sets may not belong to the same domain, such as the recently proposed evaluation setup [14] that the training set is from *mini*Imagenet [6], while the testing set is from CUB [15]. Fig. 1 shows some samples from both datasets. The source domain (*mini*Imagenet) contains pianos, dogs, steamships, shoes, and pencil-cases, while the target domain (CUB) contains different species of birds. In this case, the distributions of the training set and the testing set are significantly different, and the prior knowledge learned from *mini*Imagenet degrades on CUB. Chen et al. [14] reports that the recently proposed FSL methods [6–10,14] fail to deal with such domain generalization problem. In this paper, we refer to such a problem as **Few-shot Domain Generalization problem**. Some research fields are closely related to few-shot domain generalization. Table 1 introduces the relationship between few-shot domain generalization and some related learning paradigms.

To solve the few-shot domain generalization problem, we propose to infer categories of samples by extracting the relationship embedding with Forget-Update Module (**FUM**), which includes a channel vector sequence construction module and several forget-update modules. Channel vector sequence construction module is used to construct the relational information, and the forget-update modules are used to extract the relationship embeddings of

* Corresponding author.
E-mail address: lutong@nju.edu.cn (T. Lu).

https://doi.org/10.1016/j.patcog.2022.108704
0031-3203/

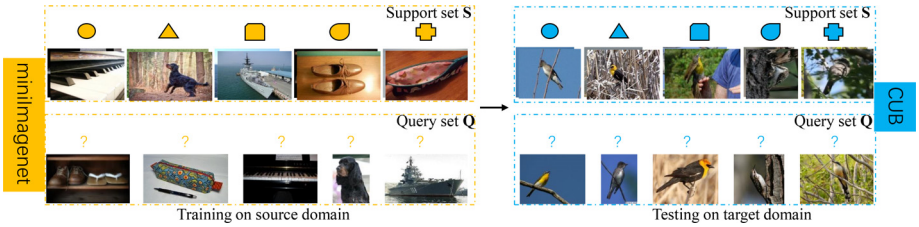
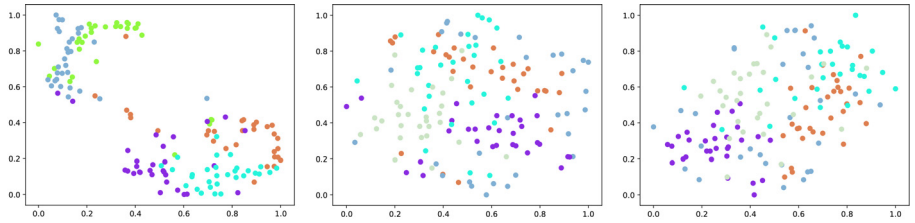


Fig. 1. Illustration for the task setting of few-shot domain generalization.

Table 1
Comparison between few-shot domain generalization and some related learning paradigms. The "FEW" column indicates whether the learning paradigm learns a model with few samples. $\mathcal{D}_{src}$ and $\mathcal{D}_{tar}$ denote the source domain set and the target domain set, while $\mathcal{Y}_{src}$ and $\mathcal{Y}_{tar}$ denote the source label set and the target label set. $P(\cdot)$ represents a distribution function.

Learning paradigm	Training data	Test data	FEW	Condition
Few-shot domain generalization	$\mathcal{D}_{src}$	$\mathcal{D}_{tar}$	Yes	$P(\mathcal{D}_{src}) \neq P(\mathcal{D}_{tar}), \mathcal{Y}_{src} \neq \mathcal{Y}_{tar}$
Few-shot learning	$\mathcal{D}_{src}$	$\mathcal{D}_{tar}$	Yes	$\mathcal{D}_{src}$ and $\mathcal{D}_{tar}$ have similar data distribution but disjoint label space
Domain adaptation	$\mathcal{D}_{src}$, $\mathcal{D}_{tar}$	$\mathcal{D}_{tar}$	No	$P(\mathcal{D}_{src}) \neq P(\mathcal{D}_{tar}), \mathcal{Y}_{src} = \mathcal{Y}_{tar}$
Domain generalization	$\mathcal{D}_{src}$	$\mathcal{D}_{tar}$	No	$P(\mathcal{D}_{src}) \neq P(\mathcal{D}_{tar}), \mathcal{Y}_{src} = \mathcal{Y}_{tar}$



(a) Relationship embeddings generated by FUM (b) Feature embeddings generated by FEAT [10] (c) Feature embeddings generated by ProtoNet [8]

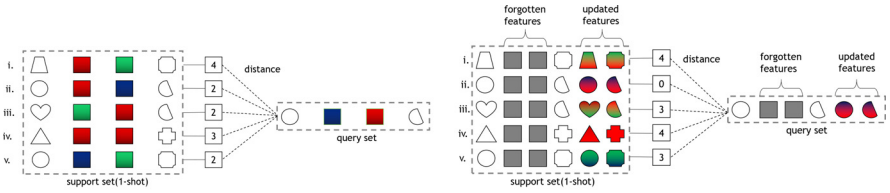
Fig. 2. Visualization of embedded features by t-SNE [16]. We use *mini*Imagenet to train different models and use these models to classify samples on the CUB dataset. Fig. 2(a)–(c) show the visualization of the embedded features generated by the different models on five types of CUB samples with t-SNE [16]. The color of the dots represents the category, and the distance between the dots represents the similarity.

channel vector sequence. The proposed method classifies few-shot samples based on the distribution of relationship embeddings. Fig. 2(b) and (c) show the distributions of feature embedding extracted by FEAT [10] and ProtoNet [8], respectively, from which we find that there are no clear boundaries between different categories. Fig. 2(a) shows the distribution of the relationship embeddings extracted by FUM. It can be seen that the relationship embeddings of FUM are more closely distributed within classes and more distantly distributed between classes than feature embeddings of FEAT and ProtoNet. Therefore, we can say that relationship embeddings processed by forgetting and updating mechanisms have better few-shot domain generalization ability than feature embeddings of FEAT and ProtoNet.

In an FSL classification task, the predicted category of a query sample is affected by samples in the support set. The motivation of channel vector sequence is to obtain the relationship embeddings. Channel vector sequence presents a simple method to exploit the related information of the support sample and a query sample by stitching their features in channel order. Relationship embeddings of a channel vector sequence is then extracted in conjunction with the forget-update module proposed in §3.3. Relationship embeddings can be used to obtain classification results in cross-domain scenarios and the same-domain scenarios.

The motivation of the proposed forget-update module is illustrated in Fig. 3. Since some distinguishing features in one task may be invalid in another task due to domain shift, the proposed forget-update module is designed to alleviate domain shifts between training and test datasets in FSL scenarios by learning to update and forget. In addition, forget-update modules extract the relationship embeddings of channel vector sequence containing the general relationship of a support set and a query sample. For each episode, we first construct a channel vector sequence, which implies the related information of the supporting samples and a query sample. Then we extract information of channel vector sequence using a series of proposed forget-update modules, which are designed to improve the discrimination by learning to forget and generate new features based on each task. The forget-update module takes into account differences between domains and adjusts the distribution of relational embeddings through a forgetting and updating mechanism based on specified tasks. In detail, a forget-update module is a task-based module that contains forgetting parts and updating parts. The forgetting part calculates the retaining ratios for the input features via the forgetting block. The partially retained features are concatenated with the newly extracted features obtained by the updating part. Finally, the concatenated result is used as input for the next forget-update step. This circular process realizes the forgetting and updating of channel vector sequence in the feature extraction process. Through continuously forgetting domain-unrelated information with the forget module and introducing domain-specific information with the update module, we finally generalize the model from base domains to novel domains, which has better few-shot domain generalization performance.

To validate the effectiveness of the proposed method, extensive experiments are conducted in both the cross-domain scenarios and the same-domain scenarios, which evaluates the learning ability of the proposed method in the cross-domain and the same-domain scenarios. The experimental results in Tables 2–9 show the supe-



(a) Initial features (b) Features processed by forget-update module

Fig. 3. A toy example illustrates the motivation of forget-update module. This is the 5-way 1-shot few-shot classification example. The support set includes five classes (i, ii, iii, iv, v), and the query set includes one sample. (Fig. 3(a)) Each class includes four features. The distances between the query sample and each supporting sample are 4, 2, 2, 3, 2. Under this condition, we cannot distinguish the category of the query sample. (Fig. 3(b)) Each sample includes six features, which include forgotten features and updated features.

Table 2
5-way 1-shot and 5-way 5-shot classification accuracies on *mini* Imagenet. The experimental results on the left were obtained without data augmentation, while those on the right with data augmentation. The experimental results marked with ◇ are from Chen et al. [14], while the experimental results marked with ♡ are from Zhang et al. [42]. The unit is a percentage. The best results are marked in **bold**, and the second-best results are marked in underline.

Model	No Data Augmentation		Data Augmentation	
	1-shot	5-shot	1-shot	5-shot
MAML [7]	47.91±0.81	62.51±0.72	46.47±0.82◇	62.71±0.71◇
MatchingNet [6]	50.53±0.83	63.77±0.67	48.14±0.78◇	63.48±0.66◇
ProtoNet [8]	48.58±0.82	64.18±0.69	44.42±0.84◇	64.24±0.72◇
RelationNet [9]	50.43±0.78	66.30±0.70	**49.31±0.85**◇	**66.60±0.69**◇
Baseline [14]	36.43±0.61	55.41±0.66	42.11±0.71◇	62.53±0.69◇
Baseline+ [14]	38.26±0.55	55.86±0.65	48.24±0.75◇	66.43±0.63◇
FEAT [10]	46.11±0.74	62.76±0.67	47.25±0.79	61.92±0.71
SS [42]	46.9 −−♡	64.0 −−♡	-	-
FUM(2,2)	**51.87±0.75**	**67.92±0.67**	47.21±0.81	65.47±0.63
FUM(2,2,2)	50.33±0.82	66.94±0.66	47.17±0.79	64.62±0.67

Table 3
5-way 1-shot and 5-way 5-shot classification accuracies on CUB. The experimental results on the left were obtained without data augmentation, while those on the right with data augmentation. The experimental results marked with ◇ are from Chen et al. [14], while the experimental results marked with ♡ are from Zhang et al. [42]. The unit is a percentage. The best results are marked in **bold**, and the second-best results are marked in underline.

Model	No Data Augmentation		Data Augmentation	
	1-shot	5-shot	1-shot	5-shot
MAML [7]	56.07±0.94	73.28±0.69	55.92±0.95◇	72.09±0.76◇
MatchingNet [6]	60.51±0.89	72.88±0.67	61.16±0.89◇	72.86±0.70◇
ProtoNet [8]	49.39±0.88	66.21±0.72	51.31±0.91◇	70.77±0.69◇
RelationNet [9]	60.83±0.92	73.82±0.67	62.45±0.98◇	76.11±0.69◇
Baseline [14]	31.95±0.58	52.90±0.67	47.12±0.74◇	64.16±0.71◇
Baseline+ [14]	43.58±0.76	60.82±0.76	60.53±0.83◇	79.34±0.61◇
FEAT [10]	52.43±0.92	66.85±0.76	63.16±0.89	81.54±0.64
SS [42]	44.1 −−♡	59.7 −−♡	-	-
FUM(2,2)	62.35±0.99	74.26±0.67	**65.40±0.95**	78.75±0.67
FUM(2,2, 2)	**62.49±0.98**	**75.66±0.69**	64.51±0.98	**82.11±0.62**

Table 4
5-way 1-shot and 5-way 5-shot classification accuracies on CUB and *mini* Imagenet→CUB. The experimental results marked with † are from Li et al. [43]. The unit is a percentage. The best results are marked in **bold**, and the second-best results are marked in underline.

Model	CUB		*mini*Imagenet→CUB	
	1-shot	5-shot	1-shot	5-shot
MAML [7]	56.07±0.94	73.28±0.69	36.03±0.68	51.65±0.75
MatchingNet [6]	60.51±0.89	72.88±0.67	38.16±0.67	51.18±0.71
ProtoNet [8]	49.39±0.88	66.21±0.72	34.16±0.60	53.05±0.74
RelationNet [9]	60.83±0.92	73.82±0.67	38.14±0.67	53.90±0.70
Baseline [14]	31.95±0.58	52.90±0.67	33.78±0.58	51.56±0.67
Baseline+ [14]	43.58±0.76	60.82±0.76	38.17±0.63	55.12±0.69
FEAT [10]	52.43±0.92	66.85±0.76	38.42±0.72	55.29±0.74
RML [43]	-	-	40.16±0.68†	56.96±0.65†
FUM(2,2)	62.35±0.99	74.26±0.67	**44.57±0.75**	56.45±0.80
FUM(2,2,2)	**62.49±0.98**	**75.66±0.69**	44.37±0.73	**60.56±0.74**

Table 5
5-way 1-shot and 5-way 5-shot classification accuracies on Real and *mini* Imagenet→Real. The unit is a percentage. The best results are marked in **bold**, and the second-best results are marked in underline.

Model	Real		*mini*Imagenet→Real	
	1-shot	5-shot	1-shot	5-shot
MAML [7]	55.40±0.93	72.94±0.72	33.97±0.71	44.57±0.76
MatchingNet [6]	55.32±0.85	74.74±0.66	50.01±0.93	64.72±0.81
ProtoNet [8]	52.81±0.88	73.70±0.67	48.13±0.90	65.01±0.81
RelationNet [9]	54.61±0.88	75.18±0.67	48.14±0.91	63.27±0.82
Baseline [14]	42.61±0.67	65.41±0.66	41.68±0.71	61.72±0.75
Baseline+ [14]	46.67±0.78	69.37±0.67	41.04±0.68	60.24±0.74
FEAT [10]	51.20±0.86	70.76±0.72	47.32±0.93	63.52±0.88
FUM(2,2)	**57.44±0.90**	**76.69±0.65**	**52.13±0.91**	68.24±0.81
FUM(2,2,2)	56.27±0.89	76.31±0.67	51.52±0.97	**68.33±0.78**

Table 6
5-way 1-shot and 5-way 5-shot classification accuracies on Painting and *mini* Imagenet→Painting. The unit is a percentage. The best results are marked in **bold**, and the second-best results are marked in underline.

Model	Painting		*mini*Imagenet→Painting	
	1-shot	5-shot	1-shot	5-shot
MAML [7]	39.48±0.78	53.15±0.75	27.34±0.55	33.38±0.57
MatchingNet [6]	38.05±0.78	51.55±0.70	35.19±0.72	46.93±0.75
ProtoNet [8]	35.80±0.70	49.10±0.68	34.63±0.71	46.70±0.72
RelationNet [9]	38.44±0.76	52.39±0.72	35.41±0.70	49.10±0.77
Baseline [14]	30.97±0.54	44.97±0.63	31.70±0.57	45.68±0.64
Baseline+ [14]	31.88±0.56	46.98±0.63	31.23±0.59	44.13±0.72
FEAT [10]	35.21±0.72	48.71±0.74	33.14±0.66	45.82±0.71
FUM(2,2)	**42.85±0.82**	**58.90±0.71**	**37.32±0.72**	50.14±0.77
FUM(2,2,2)	40.92±0.77	58.73±0.68	37.23±0.74	**50.87±0.71**

Table 7
5-way 1-shot and 5-way 5-shot classification accuracies on Infograph and *mini* Imagenet→Infograph. The unit is a percentage. The best results are marked in **bold**, and the second-best results are marked in underline.

Model	Infograph		*mini*Imagenet→Infograph	
	1-shot	5-shot	1-shot	5-shot
MAML [7]	26.45±0.62	33.53±0.62	23.22±0.47	26.22±0.49
MatchingNet [6]	26.78±0.56	32.66±0.56	26.81±0.54	32.81±0.58
ProtoNet [8]	26.41±0.53	31.36±0.56	26.85±0.54	33.45±0.60
RelationNet [9]	27.10±0.63	32.39±0.59	26.94±0.59	34.33±0.60
Baseline [14]	23.30±0.43	27.77±0.46	24.65±0.46	32.28±0.57
Baseline+ [14]	23.05±0.44	27.80±0.50	24.40±0.46	30.18±0.55
FEAT [10]	25.28±0.53	30.86±0.55	26.18±0.53	32.55±0.59
FUM(2,2)	28.24±0.63	34.12±0.63	**28.46±0.56**	33.02±0.61
FUM(2,2,2)	**28.78±0.64**	**34.35±0.62**	28.04±0.57	**35.15±0.62**

riority of our method. Tables 2 and 3 give the experimental results on the same-domain scenarios. It is worth mentioning that in Table 2, in the case of no data augmentation, FUM(2, 2) achieves the best performance, that is, 1.34% higher than the best comparison method in the 5-way 1-shot paradigm and 1.62% higher than the comparison method in the 5-way 5-shot paradigm. In Table 3, in the case of no data augmentation, FUM(2, 2, 2) also achieves the best performance, that is, 1.66% higher than the best comparison method in the 5-way 1-shot paradigm and 1.84% higher than the best comparison method in the 5-way 5-shot paradigm. Tables 4–9 give the experimental results in the same-domain and the cross-domain scenarios. Specifically, the FUM method obtains the best prediction results in almost all the cross-domain scenarios, indicating that FUM has a good few-shot domain generalization ability. In addition, the experimental results in the same-domain scenarios also show that the FUM method has an excellent few-shot learning performance.

The main contributions of this paper are shown as follows.

Table 8
5-way 1-shot and 5-way 5-shot classification accuracies on Clipart and *mini* Imagenet→Clipart. The unit is a percentage. The best results are marked in **bold**, and the second-best results are marked in underline.

Model	Clipart		*mini*Imagenet→Clipart	
	1-shot	5-shot	1-shot	5-shot
MAML [7]	46.47±0.83	65.51±0.72	29.49±0.57	38.54±0.64
MatchingNet [6]	47.23±0.82	64.24±0.69	35.10±0.69	47.66±0.69
ProtoNet [8]	44.39±0.79	62.13±0.67	34.76±0.71	48.18±0.71
RelationNet [9]	48.10±0.82	67.70±0.71	36.54±0.72	50.37±0.77
Baseline [14]	39.93±0.64	60.26±0.66	34.69±0.64	**51.96±0.69**
Baseline+ [14]	43.45±0.71	63.80±0.69	34.06±0.62	48.87±0.68
FEAT [10]	40.50±0.77	58.66±0.74	31.08±0.66	44.61±0.69
FUM(2,2)	50.00±0.79	71.00±0.63	**37.81±0.72**	48.46±0.77
FUM(2,2,2)	**50.65±0.80**	**72.27±0.63**	37.72±0.72	51.03±0.76

- This work is the first effort to perform domain generalization on few-shot learning scenarios;
- The proposed FUM presents a novel method to mitigate the bias of FSL domains and can improve the few-shot domain generalization ability by forgetting and generating features according to specific tasks;
- The proposed channel vector sequence construction module gives a new method to construct the related information for FSL scenarios by stitching channel information of samples;
- We proposed to extract relational embedding of each scenario, which considers valuable relational information between samples in a scenario. Visualization results in Fig. 2 also show that relational embedding is more discriminating than feature embedding.

2. Related work

2.1. Few-shot classification methods

In the recent years, a large number of FSL methods [7–9,17–23] have been proposed. They can be divided into the following categories: metric-based methods, meta-learning-based methods, and classifier-learning-based methods.

[8–10,17,21–23] are metric-based methods. These methods aim to make the samples from the same category closer in the embedding space, while those from different categories are further apart. For instance, Siamese Network [17] employs a siamese network to extract feature vectors from a pair of samples and calculates the similarity relationship between them. ProtoNet [8] and FEAT [10] are based on euclidean distance metrics and uses the mean of embeddings from the same category as the prototype of that category. Relation Network [9] is similar to ProtoNet, except that it employs a neural network to learn a deep instance metric instead of using a fixed one. Some metric-based methods propose to generate more robust prototypes. LMPNet [21] proposes a novel local descriptor-based multi-prototype network that generates an embedding space with multiple prototypes. UDS [22] uses a descriptor selection module to locate and select semantic regions in the feature maps and then maps the selected features into new vectors via a task-related aggregation module to enhance the representations of prototypes. [23] introduces fine-grained visual attributes that enable the meta-learner to learn to complete prototypes. Our work is related to these metric-based approaches, with the difference that we propose a few-shot domain generalization method to extract domain-adaptive relationship embedding, which considers the differences between domains and adjusts the distribution of relational embeddings by the proposed FUM.

Some meta-learner-based methods propose to construct a meta-learner that learns to make updates to the parameters of a meta-learner designed for a scenario, such as [7,18,19].

Table 9
5-way 1-shot and 5-way 5-shot classification accuracies on *mini* Imagenet→Cars, *mini* Imagenet→Dogs and *mini* Imagenet→Flowers. The unit is a percentage. The best results are marked in **bold**, and the second-best results are marked in underline.

Model	*mini*Imagenet→Cars		*mini*Imagenet→Dogs		*mini*Imagenet→Flowers	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MAML [7]	25.32±0.49	30.29±0.50	25.16±0.45	28.70±0.46	23.21±0.45	25.26±0.41
MatchingNet [6]	28.69±0.59	38.57±0.65	31.29±0.58	44.51±0.72	38.37±0.84	54.67±0.82
ProtoNet [8]	27.93±0.55	37.50±0.64	31.44±0.61	45.13±0.72	39.19±0.84	52.32±0.92
RelationNet [9]	28.17±0.59	36.70±0.61	31.39±0.63	44.29±0.69	39.94±0.84	52.10±0.84
Baseline [14]	27.32±0.49	39.78±0.59	28.11±0.50	39.26±0.60	37.14±0.67	55.20±0.78
Baseline+ [14]	27.06±0.51	37.21±0.56	30.54±0.55	41.82±0.64	37.42±0.68	53.56±0.80
FEAT [10]	27.17±0.56	35.09±0.63	28.96±0.57	42.85±0.64	36.20±0.74	49.20±0.86
FUM(2,2)	**30.76±0.61**	39.49±0.65	**35.24±0.70**	**47.50±0.70**	**42.93±0.90**	53.94±0.86
FUM(2,2,2)	30.57±0.63	**41.00±0.67**	33.05±0.66	47.04±0.70	42.46±0.87	**55.90±0.85**

MAML [7] provides a method to initialize the parameters of the traditional learner in such a way that a few gradient descent steps with a small amount of training data from a new task will lead to good generalization performance on that task[18]. proposes to use the embedding vectors of newly seen samples to imprint weights for the new classes on the rear of the base network. The traditional learner used in [19] is a convolutional-network-based network. This method learns to update the parameters of the last fully connected layer of the traditional learner on the newly seen samples.

The last type is the classifier-learning-based method. Some FSL methods are implemented by learning a classifier, such as [14,24]. These methods use a large number of sampled tasks in the training set to train a feature extractor and a classifier in the training phase, then fix parameters of the feature extractor and refine the parameters of the classifier with a few labeled samples in the testing phase. Finally, the feature extractor and the refined classifier are used to make predictions on unlabeled samples.

2.2. Domain adaptation

Domain adaptation is a particular case of transfer learning. In domain adaptation, the source and target domains are in the same feature space, but with different distributions. Domain adaptation aims to adapt the model trained in the source domain to the target domain. It has been successfully used in tasks such as style transfer and object recognition. Recently, multiple domain adaptation methods have been proposed to minimize distribution discrepancy. These include divergence-based domain adaptation, adversarial-based domain adaptation, and reconstruction-based domain adaptation. Some divergence-based domain adaptation methods [25,26] propose to minimize divergence criterion between the source and target data distributions. Some adversarial-based domain adaptation methods [27,28] propose to reduce the gap between distributions by using adversarial training. Reconstruction-based domain adaptation methods [29,30] use an auxiliary reconstruction task to create a shared representation for each domain.

These domain adaptation methods allow us to transfer the knowledge learned on source tasks to target tasks within the same category. However, they are not suitable for the FSL cross-domain scenario, where the training and testing sets have different classes. Specifically, the training and testing sets contain general object categories and different bird species, respectively. Besides, [14] reports that recently proposed transfer-learning-based FSL methods are also severely degraded in this condition. In this paper, we design forget-update modules to extract the relationship embeddings of channel vector sequence containing the general relationship between support samples and a query sample. The forget-update modules can also align the distribution of relationship embeddings by forgetting and generating feature information of channel vector sequence. The extensive experiments in Tables 4–9 demonstrate that the proposed FUM can significantly improve the performance of FSL in cross-domain scenarios.

3. Methodology

In this section, the preliminaries are given first; then the channel vector sequence is described; finally, forget-update module is detailed.

The overall framework of the proposed method is shown in Fig. 4, which consists of four parts: feature extractor $\varphi(\cdot)$, channel connector $\mathcal{C}(\cdot)$, forget-update module and prediction module. The feature extractor $\varphi(\cdot)$ first converts each image into c feature maps, and then converts each feature map into a 1-dimensional feature vector m, $m \in R^d$. Channel connector $\mathcal{C}(\cdot)$ is used to stitch the feature vector of a query sample with feature vector of each class in support set, then obtain N channel vector sequence $\widetilde{\mathbf{x}}$ ($\widetilde{\mathbf{x}} \in R^{c\times(2\times d)}$), N is the number of class in support set. These channel vector sequences are then put into forget-update modules. The proposed forget-update module is constructed with forget-update blocks, which is used to extract the relationship embedding of each channel vector sequence. The prediction module infers the class of a query sample from the relationship embedding. Finally, we calculate the mean square error loss and do backward propagation.

3.1. Preliminary

The general settings and symbols of cross-domain few-shot classification used in this paper are detailed in this section.

The purpose of few-shot domain generalization is to build a model $\Phi(\cdot)$, where the training set $\mathcal{D}_a$ and testing set $\mathcal{D}_b$ are from different domains. For example, the training set uses data from *mini*ImageNet [6] that is mainly generic objects while the testing set uses data from Caltech-UCSD Birds-200-2011 (**CUB**) [15], Oxford-Flowers-102 (**Flowers**) [31], Stanford-dogs (**Dogs**) [32], Stanfords-cars (**Cars**) [33] or four datasets (**Real, Painting, Infograph, Clipart**) from DomainNet Dataset [34].

An effective way to utilize the training set is to mimic the few-shot learning setting proposed in [6]. Concretely, in the training phase, data is randomly sampled from the training set $\mathcal{D}_a$ to simulate a test scenario that is called a task (or episode) $\mathcal{T}$. The FSL model $\Phi(\cdot)$ is trained with randomly sampled tasks. Each FSL task $\mathcal{T}$ contains a support set $\mathcal{S}$, a query set $\mathcal{Q}$, and an output set $\mathcal{Y}$, which satisfy that the elements of $\mathcal{S}$ and $\mathcal{Q}$ do not intersect but are sampled from the same categories. Besides, each element in the support set $\mathcal{S}$ has label information, while those in the query set $\mathcal{Q}$ do not. The element of the output set $\mathcal{Y}$ is the label of the corresponding element in the query set $\mathcal{Q}$. In the testing phase,

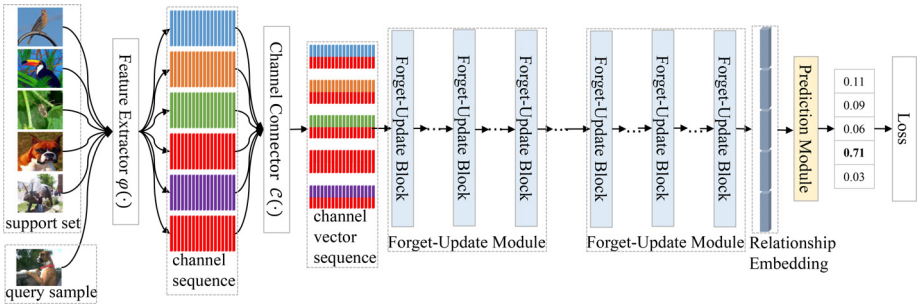


Fig. 4. The overall framework of the proposed method.

many tasks are randomly sampled from the dataset $\mathcal{D}_b$, and evaluated on the model $\Phi(\cdot)$. The average prediction accuracy of the model is used to measure the learning ability in cross-domain scenarios.

In this paper, the N-way K-shot few-shot classification paradigm is only considered unless otherwise specified. In such a paradigm, every support set $\mathcal{S}$ contains exactly N classes, and each class has K samples, while the query set $\mathcal{Q}$ contains some unlabeled samples that belong to the classes in $\mathcal{S}$. The output set $\mathcal{Y}$ includes the corresponding labels of elements in the query set $\mathcal{Q}$. The support set $\mathcal{S}$, the query set $\mathcal{Q}$ and the output set $\mathcal{Y}$ are formalized as shown in Eqs. (1), (2) and (3).

$$\mathcal{S} = \{(\mathbf{x}_{11}, l_1), \cdots, (\mathbf{x}_{ij}, l_i), \cdots, (\mathbf{x}_{NK}, l_N); \\ l_i \in \{1, \cdots, N\}\}, \tag{1}$$

$$\mathcal{Q} = \{\dot{\mathbf{x}}_1, \cdots, \dot{\mathbf{x}}_q\}, \tag{2}$$

$$\mathcal{Y} = (y_1, \cdots, y_q) \in \{1, \cdots, N\}^q, \tag{3}$$

where N is the number of classes, K is the number of samples per class in the support set $\mathcal{S}$, and q is the size of the query set. The subscripts of $\mathbf{x}_{ij}$ indicate that $\mathbf{x}_{ij}$ is the jth sample in the ith class.

Meta-learner $\Phi(\cdot)$ is trained to fit few-shot classification tasks by minimizing the prediction loss on query set $\mathcal{Q}$ as in Eq. (4).

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathbb{E}_{\mathcal{T}} \left[\sum_{\dot{\mathbf{x}}_i \in \mathcal{Q}, y_i \in \mathcal{Y}} \mathcal{L}(y_i, \Phi_\theta(\dot{\mathbf{x}}_i, \mathcal{S})) \right], \tag{4}$$

where $\Phi(\cdot)$ indicates the meta-learner and $\mathcal{L}(\cdot)$ is the loss function. The meta-learner $\Phi(\cdot)$ is required to train with thousands of randomly sampled tasks under the constraint of a loss function $\mathcal{L}(\cdot)$.

3.2. Channel vector sequence construction module

In this paper, channel vector sequence is proposed, which is constructed to collect the related information of each class-level feature map and a query sample feature in a task. Channel vector sequence can be combined with some sequence prediction methods, such as TCN, to predict the similarity between the query sample and each supported class based on a specific task.

This part introduces how to convert samples in a task $\mathcal{T}$ to a channel vector sequence $\widetilde{\mathbf{x}}_p$. First, the feature extractor $\varphi(\cdot)$ is used to extract feature maps of samples in support set $\mathcal{S}$ and query set $\mathcal{Q}$. The feature extractor can be implemented using a deep convolutional neural network. It yields a feature map tensor which has a dimension of (c, h, w), where c, h and w indicate dimensions of channel, height and width. Class-level feature maps are then obtained by averaging the feature maps of the same class in the support set $\mathcal{S}$. The class-level feature map for the ith class in support set $\mathcal{S}$ is formulated as follows:

$$\overline{\mathbf{x}}_i = \frac{1}{K} \sum_{j=1}^{K} (\varphi(\mathbf{x}_{ij})). \tag{5}$$

Next, each class-level feature map $\overline{\mathbf{x}}$ and feature map of each query sample $\varphi(\dot{\mathbf{x}})$ are converted to feature vector $\overline{\mathbf{x}}^{'}$ and $\dot{\mathbf{x}}^{'}$ with a dimension of $(c, h \times w)$. After that, the dimensions of $\overline{\mathbf{x}}^{'}$ and $\dot{\mathbf{x}}^{'}$ are changed to (c, d) using a mapping function. We refer to the results in Table 12 to choose d and the mapping function. Finally, the channel connector $\mathcal{C}(\cdot)$ splices the ith class-level feature vector $\overline{\mathbf{x}}_i^{'}$ with the feature vector of the pth query sample $\dot{\mathbf{x}}_p^{'}$ according to the channel order to form channel vector sequence $\widetilde{\mathbf{x}}$, as Eq. (6).

$$\widetilde{\mathbf{x}}_{ip} = \mathcal{C}(\overline{\mathbf{x}}_i^{'}, \dot{\mathbf{x}}_p^{'}). \tag{6}$$

where $\mathcal{C}(\cdot)$ is the channel connector function, p indicates pth element in query set $\mathcal{Q}$. $\widetilde{\mathbf{x}}_{ip} \in \mathbb{R}^{c \times (2 \times d)}$, c is the number of channels, and d is the feature dimension of each reduced instance. $\widetilde{\mathbf{x}}$ is called **channel vector sequence** in this paper.

Then, the few-shot classification problem is transformed into a sequence prediction problem on channel vector sequence. The prediction model is formalized as Eq. (7).

$$\widetilde{y}_p = f(\widetilde{x}_{1p}, \cdots, \widetilde{x}_{Np}), \tag{7}$$

where p indicates pth element in query set $\mathcal{Q}$, N is the number of classes in a support set and $f(\cdot)$ is a prediction model.

Table 10
Ablation experiments on *mini* Imagenet→CUB and CUB. The unit is a percentage and the best results are highlighted .

Model	*mini*Imagenet→CUB		CUB	
	1-shot	5-shot	1-shot	5-shot
+forget	43.56±0.19	54.45±0.19	59.94±0.23	72.25±0.18
+update	43.76±0.19	58.20±0.18	61.73±0.23	74.56±0.17
FUM(2,2,2)	**44.30±0.19**	**60.44±0.18**		
	61.97±0.23	**75.51±0.17**		

3.3. Forget-Update module

Most state-of-the-art FSL methods do not consider the domain shift problem between the training and testing sets. These methods may degrade when there is a domain shift between them. In this section, forget-update module is proposed for improving the discrimination in the scenario of domain shift. Forget-update module consists of stacked forget-update blocks, and each forget-update block consists of forgetting block and updating block. The forgetting block is designed to learn the forgetting rate based on context, while the updating block learn to generate new information according to context. By training on numerous scenarios, the forget-update module learns how to forget noisy information that does not fit the context and generate new information based on

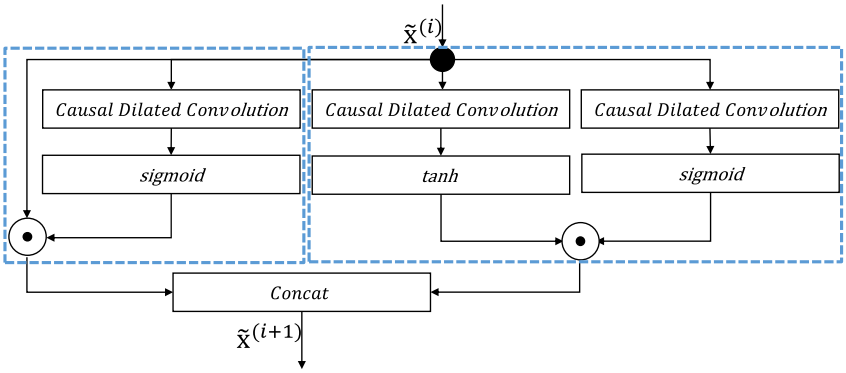


Fig. 5. Forget-update block. The forget-update block includes two parts. The dashed boxes from left to right represent the forgetting block and updating block, respectively. $\odot$ indicates element-wise multiplication.

the context. With the help of forget-update module, we can extract the relationship embeddings from channel vector sequence, which helps to make the distribution of the training and testing sets consistent. As can be seen in Fig. 2, the distribution based on relationship embeddings (Fig. 2(a)) is more distinguishable than the instance-embedding-based distribution (Fig. 2(b) and (c)). Forget-update module can bring similar samples closer together and different categories further apart. The overall flow of forget-update module described in Algorithm 1 and the detail of forget-update block illustrated in Fig. 5.

Algorithm 1 Forget-update module. $\sigma(\cdot)$ is the sigmoid function, $\odot$ denotes element-wise multiplication operator, and $\mathcal{C}(\cdot)$ is the channel concatenate function, $Causal(\cdot)$ is the causal dilated convolutional function.

Require:
$\tilde{\mathbf{x}}$: channel vector sequence
k: kernel size
c: the length of channel sequence

$\tilde{\mathbf{x}}^{(0)} = \tilde{\mathbf{x}}$
$\ell = \lceil \log_k c \rceil$
$i = 0$
while $i < \ell$ **do**
  $d = k^i$
  $\tilde{\mathbf{x}}^{(i+1)} = \text{ForgetUpdateBlock}(\tilde{\mathbf{x}}^i, d, k)$
  $i = i + 1$
end while
return $\tilde{\mathbf{x}}^{(\ell-1)}$

Causal dilated convolution is the basis of forget-update block. Causal dilated convolution is a special case of standard dilated convolution. It is first applied as a special one-dimensional convolution in Wavenet [35], which can be implemented by shifting the output of a normal convolution by a few steps. For two-dimensional data, the equivalent of causal convolution is PixelCNN [36]. When combining the casual convolution with dilated convolution, the network can produce outputs of the same length as the inputs. It can obtain features as data leakage-free with few network layers. It can be formalized as Eq. (8). Dilated convolution is adopted to improve the range of receptive field on the channel vector sequence. The dilation factor d increases exponentially and can be formalized as Eq. (9).

$$r = Causal(\tilde{\mathbf{x}}, d, k), \tag{8}$$

$$d = k^{\ell}, \tag{9}$$

where $\tilde{\mathbf{x}}$ is the input of causal dilated convolution, d indicates dilated rate, k denotes kernel size, and ℓ indicates the ℓth layer.

The proposed forgetting block learns how to forget low-recognition features based on the context. In the meta-training process, the feature extractor gathers critical training experiences, while the forgetting block is designed to discard redundant and deprecated information and concentrates our attention upon the most relevant and critical pieces of information by calculating the retaining ratio of the input features. Specifically, the initial context of forgetting block is channel vector sequence, and all subsequent contexts are the output of the previous forget-update block. Forgetting block implements the forgetting mechanism by calculating the forgetting rate of the input sequence. The forgetting block generates data $\mathbf{x}_{forget}(\mathbf{x}_{forget} \in \mathbb{R}^{c \times d_{in}})$ of the same size as the input, which can be formalized as Eq. (10).

$$\mathbf{x}_{forget} = \sigma(Causal(\tilde{\mathbf{x}}^{(i)}, d, k)) \odot \tilde{\mathbf{x}}^{(i)}, \tag{10}$$

where $Causal(\cdot)$ is causal dilated convolutional function, $\sigma(\cdot)$ is a sigmoid function, d is dilated rate, k is kernel size, $\tilde{\mathbf{x}}^{(i)}$ is the input to the ith forget-update block in forget-update module, $\odot$ indicates element-wise multiplication.

The proposed updating block is designed to learn and generate information from the combination feature of a query instance and support instances in the meta-training process, where the information are extracted from multiple instances and can be used as a complement to the feature information extracted by the feature extractor $\varphi(\cdot)$. In addition, each forget-update module contains multiple update blocks that can generate a number of different levels of information. Specifically, the channel vector sequence $\tilde{\mathbf{x}}$ is used as the initial context of the first updating block, and the rest of the contextual information is the output from the previous layer. Updating block generates data $\mathbf{x}_{update}(\mathbf{x}_{update} \in \mathbb{R}^{c \times filter\_size})$ with the same sequence length as the input, which can be formalized as Eq. (11).

$$\mathbf{x}_{update} = \tanh(Causal(\tilde{\mathbf{x}}^{(i)}, d, k))) \odot \sigma(Causal(\tilde{\mathbf{x}}^{(i)}, d, k))), \tag{11}$$

where $tanh(\cdot)$ is hyperbolic tangent activation function.

Finally, the stitched $\mathbf{x}_{forget}$ and $\mathbf{x}_{update}$ are used as the output of forget-update block.

4. Experiments

The validity of the proposed method is evaluated in this section. For the sake of fairness of the experiments, a uniform experimental platform provided by Chen et al. [14] is used to conduct comparison experiments.

4.1. Experimental setups

All methods are trained from scratch unless otherwise specified. Adam [37] is used as the optimizer.

The optimizer takes an initial learning rate of 0.001, and the learning rate is reduced by 10% when the accuracy verified on the validation set stagnates in seven consecutive training steps. The most common FSL classification settings, 5-way 1-shot and 5-way 5-shot classification, have experimented on all the datasets. Unless otherwise noted, all results are averaged over 600 episodes from the testing set with a 95% confidence interval. The detailed setup of scenarios, network architecture, training schema, and evaluation protocols are as follows.

4.1.1. Scenarios

Two experimental settings are selected for testing. The first one is the same-domain scenario, where the training and testing sets are all selected from the same-domain. The second one is a cross-domain scenario, where the training and testing sets are selected from *mini*Imagenet [6] and other domains, respectively.

For the same-domain scenario, the training set, validation set and test set are from the same domain. These same-domain scenarios are mainly used to test few-shot learning ability. In this paper, *mini***Imagenet** [6], Caltech-UCSD Birds-200-2011 (**CUB**) [15], Oxford-Flowers-102 (**Flowers**) [31], Stanford-dogs (**Dogs**) [32], Stanfords-cars (**Cars**) [33] and four datasets (**Real, Painting, Infograph, Clipart**) from DomainNet Dataset [34] are used in our experiments.

For the cross-domain scenario, 64 *mini*Imagenet training classes are used as the training set, 16 *mini*Imagenet validation classes are used as the validation set, and the trained model is tested on each test set from different domains. Cross-domain scenarios are mainly used to test few-shot domain generalization capabilities. These scenarios include: *mini***Imagenet→CUB**, *mini***Imagenet→Real**, *mini***Imagenet→Paintting**, *mini***Imagenet→Inforgraph**, *mini***Imagenet→Clipart**, *mini***Imagenet→Cars**, *mini***Imagenet→Dogs** and *mini***Imagenet→Flowers**.

The details of these datasets are as follows:

- *mini*Imagenet dataset is a subset of Imagenet [38] and consists of 100 generic object classes, each of which contains 600 images. Follow the standard protocol [6] that the dataset is split into 64, 16, and 20 classes for training, validation, and testing, respectively;
- CUB [15] dataset contains 11,788 images from 200 species of birds in total, which is commonly used for fine-grained classification. There is little difference in domains between CUB categories. Following the commonly used evaluation protocol [14,39], the dataset is split into 100, 50, and 50 classes for training, validation, and testing, respectively;
- DomainNet Dataset [34] is the largest unsupervised domain adaptation dataset to date, which contains six domains and about 600,000 images distributed in 345 categories. We report experimental performance in the following domains: **Clipart**, the collection of clipart images; **Infograph**, the collection of infographic images with specific object; **Painting**, the collection of artistic depictions in the form of painting; **Real**, images collected in the real world. DomainNet Dataset is used to validate the few-shot learning ability and the few-shot domain generalization ability of each model in this paper. To evaluate the few-shot domain generalization ability of each model, we train and evaluate the model using the training and validation sets of *mini*Imagenet, respectively, and calculate the prediction accuracy of the model using the test set of DomainNet Dataset. To evaluate the few-shot learning ability, we divide each domain of DomainNet Dataset into three disjoint parts, where the number of classes in the training set, validation set, and test set is 300, 15, and 30, respectively.
- Flowers dataset [31] is a collection of 102 species from common flowers, and each category contains 40 to 258 images. These images have rich variations in proportions, poses, and lighting. We test the cross-domain generalization ability of all the models on the test set, including 102 categories;
- Dogs dataset [32] is constructed using ImageNet images and annotations, and it contains 20, 580 images of 120 breeds of dogs from all over the world, which is commonly used for fine-grained classification. We test the cross-domain generalization ability of all the models on the test set, including 120 categories;
- Cars dataset [33] contains 16,185 images from 196 classes of cars, which is often used for fine-grained classification. We test the cross-domain generalization ability of all the models on the test set, including 196 categories.

4.1.2. Network architecture

For a fair comparison, a four-layer convolutional backbone (Conv-4) is used, as in [6–9,14], which consists of four blocks and each block outputs 64 channels. The input size of images is 84×84.

The proposed model FUM(2,2) and FUM(2,2,2) contains two and three forget-update modules, respectively. Each forget-update module contains $\lceil \log_k c \rceil$ $(k = 2, c = 64)$ forget-update blocks. The filter_size of forget-update blocks in each forget-update module is set to 2 according to the sensitivity experiments in §4.4.

The prediction module is a one-layer fully connected network. The weights of the prediction model are initialized with [40] and normalized with [41]. The prediction module predicts N values representing the similarity between the query sample and the N classes in the support set.

4.1.3. Training schema

Normalization operation is applied to the input images. The proposed method is trained with an episodic training strategy [6], which is considered as a promising direction in handling the challenge of learning transformable visual concepts with limited annotations. In each episode, N classes are randomly selected. Then, for each chosen class, K labeled images are randomly selected to form the support set, and 16 images are selected from each of the remaining samples of these N classes to form the query set.

4.1.4. Evaluation protocols

The performance of FSL methods are evaluated on validation classes, and only the model which has the best performance is saved. The 5-way 1-shot and 5-way 5shot settings are performed, and 15 query samples are selected from each class in an episode. The testing accuracy is the average accuracy (%, top-1) of all the prediction results of 600 episodes sampled from the testing set with a 95% confidence interval.

4.2. Experimental results

To ensure a fair comparison of all the methods, a unified testbed is adopted for few-shot classification algorithms provide by Chen et al. [14]. Tables 2, 3 show the experimental results on *mini*Imagenet and CUB, respectively. Tables 4–8 show the experimental results for different datasets under the same scenarios and cross-domain scenarios. Table 9 shows the experimental results in three cross-domain scenarios.

The experimental results in Tables 4–8 show that the domain shift between the training and testing set can significantly affect

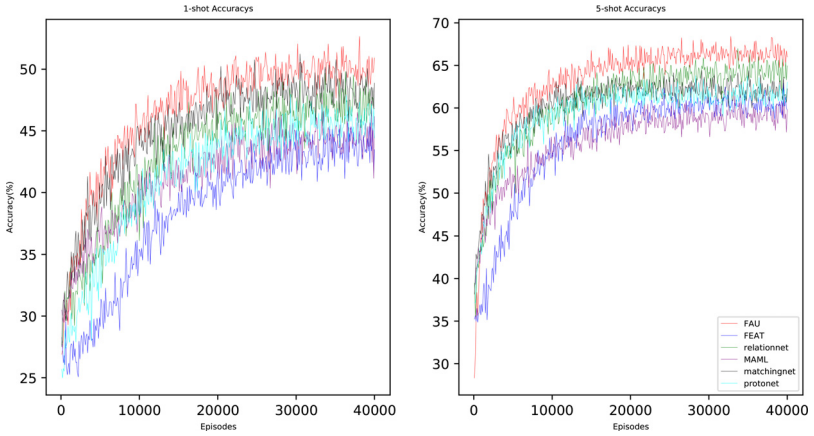


Fig. 6. 5-way 1-shot (left) and 5-way 5-shot (right) accuracy on *mini* Imagenet without data augmentation.

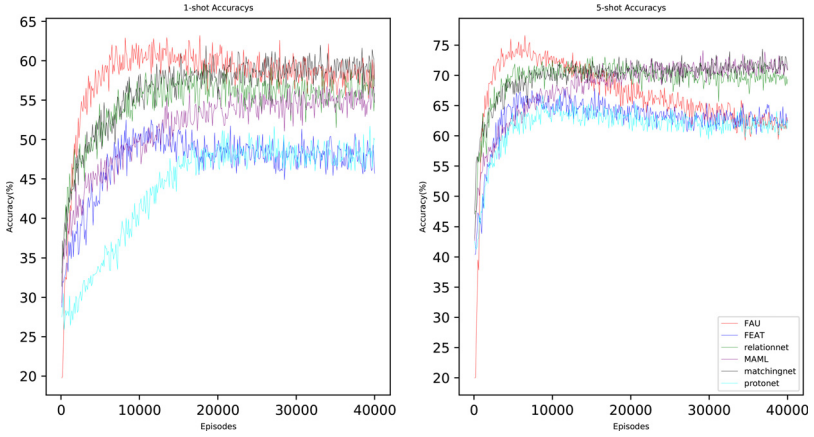


Fig. 7. 5-way 1-shot (left) and 5-way 5-shot (right) accuracy on CUB without data augmentation.

FSL methods. The experimental results on *mini*Imagenet→CUB decrease significantly compared to CUB. For example, the accuracy of MatchingNet decrease by 26.51% at 1-shot and 37.34% at 5-shot. The accuracies of other methods also decreases significantly.

4.2.1. Experimental results in the same-domain scenarios

To validate the capability of the proposed FUM method in the field of few-shot learning, Tables 2, 3 demonstrates the experimental results on *mini*Imagenet and CUB dataset, respectively, and the left part of Tables 5–8 show the experimental results on four domains. In these experiments, the training and testing sets come from the same-domain.

Tables 2, 3 show that, without data augmentation, the proposed method can obtain the optimal prediction results. Also in Tables 5–8, the proposed method achieve the optimal prediction accuracy in the same-domain scenarios. In particular, FUM(2,2) is 1.34% and 1.62% higher in the 1-shot and 5-shot settings compared to the suboptimal method on *mini*Imagenet dataset. In Tables 2, 3, the same data augmentation methods are used in the meta-training phase as in [14], including random crop, left-right flip, and color jitter. Table 3 shows that FUM(2,2,2) yields SOTA results in 1-shot and 5-shot settings.

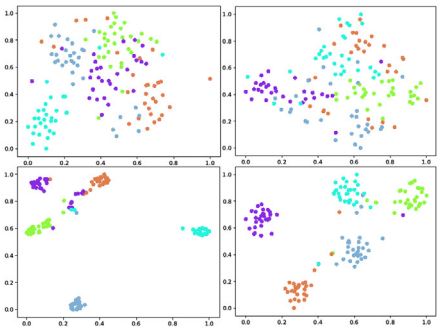
In addition, Figs. 6, 7 show how the prediction accuracy varies with the training progress on the *mini*imagenet and CUB without data augmentation, which also demonstrates that FUM(2,2) can achieve the best accuracy on *mini*Imagenet and CUB.

Although the training and test sets are in the same domain, the training and test set categories are disjoint. The learned relationship embeddings by FUM can also benefit the same-domain datasets.

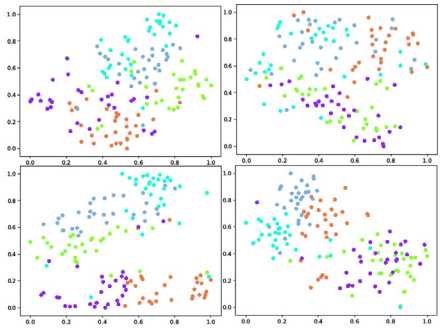
4.2.2. Cross-domain adaptive capacity

To validate the few-shot domain generalization capacity of the proposed FUM, experiments are conducted on *mini*Imagenet →CUB, *mini*Imagenet →Real, *mini*Imagenet →Painting, *mini*Imagenet → Infograph, *mini*Imagenet → Clipart, *mini*Imagenet → Cars, *mini*Imagenet → Dogs and *mini*Imagenet → Flowers in Tables 4–9, where the training set is *mini*Imagenet while the testing sets are from other domains. Experimental results show that, FUM can achieve better results than all comparison methods on *mini*Imagenet → CUB, *mini*Imagenet → Real, *mini*Imagenet→ Painting, *mini*Imagenet → Infograph, *mini*Imagenet → Cars, *mini*Imagenet → Dogs and *mini*Imagenet → Flowers. On the *mini*Imagenet→Clipart, the proposed method also achieves the optimal solution in the 5-way 1-shot paradigm, and the suboptimal solution is achieved in the 5-way 5-shot paradigm. Specifically, the proposed model is 2.12% higher than the best compare method in the 5-way 1-shot paradigm and 3.32% higher in the 5-way 5-shot paradigm on *mini*Imagenet → Real.

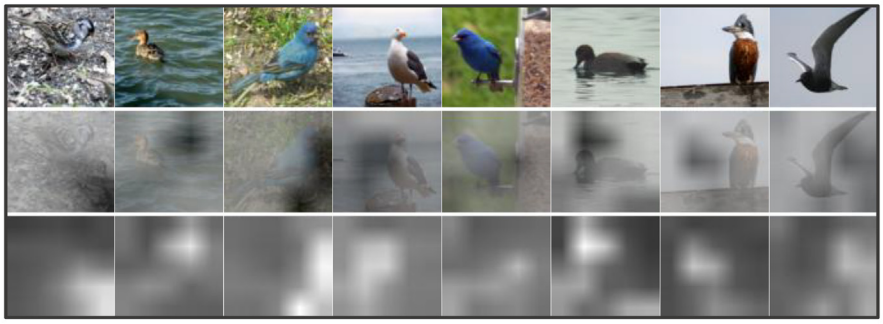
Fig. 2 (a) demonstrates the distribution of relationship embeddings learned by FUM on *mini*Imagenet→CUB. Compared to Fig. 2(b) and (c), the distribution of relationship embeddings obtained by FUM is closer between the same categories and farther



(a) The impact of forget module (left part) and update module (right part) on CUB



(b) The impact of forget module (left part) and update module (right) on *mini*Imagenet→CUB



(c) Visualization of forgetting effects

Fig. 8. The effect of forget and update modules. Fig. 8(a) and (b) show the distributions of features on CUB and *mini*Imagenet→CUB. Specifically, the top half shows the feature distribution of the backbone output, while the bottom half corresponds from left to right to the feature distributions of the forget module and update module outputs. Fig. 8(a) and (b) are visualized using t-SNE [16]. Fig. 8(c) visualizes the effect of the forget module. The first row is the original image. The third row is the visualization of the forgotten information, where the forgetting information is the result of subtracting the output from the input of the forgetting module. The second row is the result of fusing the forgotten information with the original image.

between the different categories. The relationship embeddings extracted by FUM is more suitable for the FSL classification problem in cross-domain scenario. This is because FUM can align the distribution of relationship embeddings in different domains by updating and forgetting mechanisms.

4.3. Ablation experiments

4.3.1. The effects of forgetting block and updating block

The effects of forgetting block and updating block are evaluated in this section. The results of the ablation experiments are shown in Table 10. The testing accuracy is the average accuracy (%, top-1) of all the prediction results of 10,000 episodes sampled from the testing set with a 95% confidence interval. **+update** method and **+forget** method use the same configuration as **FUM(2,2,2)** method, the only difference is that +forget method only use the forgetting block while +update only use updating block. The forgetting block is shown in the left dashed box of Fig. 5 while the updating block is shown in the right. The FUM(2,2,2) method uses forget-update blocks. All these three methods put channel vector sequence as input. Table 10 shows that when the forgetting block and updating block are combined, they can improve the prediction performance on *mini*Imagenet→CUB and CUB. The experimental results of FUM(2,2,2) are better than using a single module. Specially, our method improves by 5.99% relative to +forget and 2.24% relative to +update on the *mini*Imagenet→CUB in 5-way 5-shot paradigm, and our method improves 3.26% relative to +forget and 0.95% relative to +update method on the CUB dataset in 5-way 5-shot paradigm. The prediction results of our method are also improved in the 5-way 1-shot paradigm.

4.3.2. Visualization and analysis of forget module and update module

To analyze the effectiveness of forget module and update module, Fig. 8(a) and (b) visualize the feature distribution of the output of forget module and update module, respectively, using t-SNE [16]. And Fig. 8(c) visualizes the features forgotten by the forget module.

Specifically, the left half of Fig. 8(a) visualizes the distributions of features on CUB before and after forget module. Features of the same category output by the forget module can be better clustered together, while the distance between features of different categories is greater. The left half of Fig. 8(b) visualizes the distributions of features on *mini*Imagenet→CUB before and after forget module, which also shows that the forget module enables better discriminability between different categories of images.

The right half of Fig. 8(a) visualizes the distribution of features on CUB before and after the update module, which shows that features acquire greater discriminability after using the features generated by the update module. The right half of Fig. 8(b) visualizes the distribution of features on *mini*Imagenet→CUB before and after the update module, and the updated features also gain better discriminability. With the above visualization results, we believe that the forget module and update module can produce more meaningful features for both the same-domain and the cross-domain scenarios.

Fig. 8 (c) visualizes the forgotten features. To visualize the effect of the forget module, only the forget modules are used in this experiment. The forgotten features are defined as the input features of the first forget module minus the output of the last forget module. Then, the forgotten features are converted to a gray image of a signal channel of size 5*5 and resized to the same size as

Table 11
Sensitivity analysis of forget-update module number and filter size. The top results are highlighted and the unit is a percentage.

Model	*mini*Imagenet→CUB		CUB		*mini*Imagenet	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
FUM(2)	43.64±0.76	56.80±0.76	62.40±0.92	73.78±0.67	51.26±0.81	66.01±0.67
FUM(4)	43.15±0.76	58.15±0.72	62.97±0.96	74.68±0.72	51.53±0.81	66.42±0.65
FUM(8)	43.69±0.79	55.77±0.79	62.40±0.93	74.67±0.74	51.23±0.82	67.40±0.63
FUM(16)	43.46±0.75	52.95±0.76	61.50±0.98	73.43±0.70	50.35±0.77	64.88±0.64
FUM(32)	45.15±0.77	54.62±0.81	62.89±0.94	74.56±0.72	50.39±0.79	66.18±0.63
FUM(64)	43.79±0.78	54.81±0.71	60.98±0.96	74.22±0.71	50.22±0.79	64.24±0.69
FUM(2,2)	44.57±0.75	56.45±0.80	62.35±0.99	74.26±0.67	**51.87±0.75**	**67.89±0.67**
FUM(4,4)	44.48±0.75	54.02±0.77	61.90±0.91	74.18±0.72	50.33±0.82	66.66±0.65
FUM(8,8)	44.45±0.75	55.57±0.74	62.81±0.95	74.56±0.69	50.36±0.79	66.39±0.69
FUM(16,16)	43.81±0.76	56.65±0.73	63.44±0.96	73.76±0.73	51.40±0.79	65.23±0.64
FUM(32,32)	44.88±0.75	58.18±0.75	61.14±0.94	74.09±0.76	51.36±0.76	65.07±0.66
FUM(64,64)	44.72±0.77	52.36±0.75	61.72±0.96	74.81±0.70	50.19±0.82	65.02±0.66
FUM(2,2,2)	44.37±0.73	**60.56±0.74**	62.49±0.98	75.66±0.98	50.33±0.82	66.94±0.66
FUM(4,4,4)	46.13±0.77	56.72±0.81	63.42±0.93	**76.82±0.68**	50.84±0.84	66.16±0.66
FUM(8,8,8)	**46.64±0.75**	58.08±0.71	**64.03±0.99**	75.86±0.67	50.17±0.75	66.68±0.66

Table 12
Sensitive experiments on the instance dimension *d*. The top results are highlighted and the unit is a percentage. 25* indicates using the identity mapping function.

d	*mini*Imagenet→CUB	
	1-shot	5-shot
8	39.25±0.73	53.10±0.67
16	40.53±0.74	56.16±0.66
25	41.61±0.70	55.23±0.69
32	41.92±0.75	55.37±0.69
64	41.26±0.73	52.64±0.70
25*	**44.37±0.73**	**60.56±0.74**

the original image. Specifically, the original images are in the first row; the forgotten features are in the third row, where the white areas represent forgotten features; the second row is the weighted fusion of the original image and the forgotten features. In detail, the weight of the original image is 0.2, and the weight of the forgotten features is 0.8. To visualize the forgotten regions, we preprocessed the forgotten features by subtracting the forgotten features with 255 so that in the second row, the black area represents the forgotten part. It can be found that the forget module can forget some background information. We think that forget some irrelevant background information can help improve features, which is also supported by the feature distributions of forget module in Fig. 8(a) and (b).

4.4. Sensitivity analysis of forget-update module number and filter size

To analyze parameter sensitivity of forget-update module number and filter size, a batch of experiments is organized in Table 11. In Table 11, the numbers in parentheses after FUM correspond to the filter size in the forget-update module, and the number of numbers indicates the number of forget-update modules included in the model. For example, FUM(4,4) indicates that the model contains two forget-update modules, each of which has a filter size of 4. Table 11 demonstrates that the optimal results are obtained on CUB and *mini*Imagenet→CUB when the number of forget-update modules is equal to 3, and on *mini*Imagenet when the number of forget-update modules is equal to 2 and the filter size is equal to 2. In this paper, FUM(2,2) and FUM(2,2,2) are chosen in the paper.

4.5. Sensitive experiments on the instance dimension d

The proposed method adopts four layers convolutional network to extract feature maps, which converts a 3-channel 84*84 instance to a 64-channel 5*5 feature map. Then each feature map is transformed into a 25-dimensional vector. We try to apply a 1-layer fully connected network or an identity mapping function to the vector and generate a *d* dimensional vector. To analyze the effect of different *d* values on the prediction accuracy, Table 12 shows the prediction accuracy of the proposed method on *mini*Imagenet→CUB when different *d* values are used. We find that the best prediction accuracy is obtained when using the identity mapping function, and this configuration is exploited in all experiments.

4.6. The effective of channel vector sequence

After the channel vector sequence is generated, the rest to do is to extract the internal relationship of the channels and use such information to perform a few-shot classification. Table 13 shows that the combination of TCN with the proposed channel vector sequence can get competitive results on *mini*Imagenet→CUB and CUB. Moreover, the FUM(2,2,2) approach achieves better results than TCN at similar model sizes, implying that the forgetting and updating mechanism of forget-update module is more suitable for the proposed channel vector sequence than TCN.

4.7. Computational expense analysis

Table 14 shows the computational expense and prediction accuracy of the most commonly used FSL methods on *mini*Imagenet→Painting scenario in 5-way 1-shot paradigm, where **MAC** represents Memory Access Cost, **Params** represents the size of model parameters, **Accs** represents the prediction accuracy, and **Time** represents the time used to predict an episode. All experiments were conducted using an NVIDIA GeForce GTX 1080Ti to test. For MAC values, MatchingNet, ProtoNet, FEAT, and our method have similar MAC, Baseline, and Baseline++ methods have lower MAC, and RealtionNet method has much higher MAC. In terms of the number of parameters, ProtoNet, Baseline, and FEAT have fewer parameters, while RelationNet and Baseline++ use more parameters than the proposed methods. Baseline and Baseline++ are faster, while the rest of the methods are relatively close in terms of execution time. In terms of prediction accuracy, our method has a more obvious advantage. Comparing the proposed FUM(2,2)

Table 13
The prediction results of the TCN and FUM network. The unit is a percentage, and the best results are highlighted.

Model	Model Size	*mini*Imagenet→CUB		CUB	
		1-shot	5-shot	1-shot	5-shot
TCN [44]	1170K	39.95±0.74	55.36±0.73	60.42±0.96	74.59±0.74
FUM(2,2,2)	1190K	**44.37±0.73**	**60.56±0.74**	**62.49±0.98**	**75.66±0.98**

Table 14
The computational expense and accuracy on *mini* Imagenet→Painting scenario with 5-way 1-shot paradigm .

Model	MAC	Params	Accs	Time
MatchingNet [6]	33.667G	342.400K	35.19±0.72%	0.37s
ProtoNet [8]	33.647G	226.176K	33.99±0.68%	0.37s
RelationNet [9]	68.966G	452.753K	35.41±0.70%	0.15s
Baseline [14]	6.334G	226.176K	31.70±0.57%	0.05s
Baseline+ [14]	6.339G	546.376K	31.23±0.59%	0.06s
FEAT [10]	33.647G	242.624K	33.14±0.66%	0.28s
FUM(2,2)	36.581G	324.192K	37.32±0.72%	0.21s
FUM(2,2,2)	38.971G	404.036K	37.23±0.74%	0.46s

Table 15
Compare causal dilation convolution with fixed dilation rate convolution (named Fixed-d). The unit is a percentage. The best results are highlighted.

Model	*mini*Imagenet→CUB		*mini*Imagenet	
	1-shot	5-shot	1-shot	5-shot
Fixed-d	42.67±0.74	57.40±0.74	49.82±0.82	65.24±0.68
FUM(4)	**43.15±0.76**	**58.15±0.72**	**51.26±0.81**	**66.01±0.67**

Table 16
Impact of pre-trained model on few-shot domain generalization tasks. FUM(4,4,4) and FUM(4,4,4)_fix use the same configuration, except that the backbone network of FUM(4,4,4)_fix uses the parameters pre-trained on the training set of *mini*Imagenet and fixed these parameters.

Model	*mini*Imagenet		*mini*Imagenet→Real		*mini*Imagenet→Painting	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
FUM(2,2,2)	50.33±0.82	66.94±0.66	51.52±0.97	68.33±0.78	37.23±0.74	50.87±0.71
FUM(2,2,2)_fix	53.38±0.78	65.08±0.67	52.96±0.92	64.82±0.80	38.31±0.74	48.49±0.78

method with the two comparison methods with the highest prediction accuracy, it is found that the proposed FUM(2,2) has higher accuracy when the MAC and Params metrics are close to or smaller than the two comparison methods, RealtionNet and MatchingNet. Compared with Baseline and Baseline++, the proposed method is higher in MAC, Params, and Time. However, the advantage of our method in prediction accuracy is more obvious.

4.8. Compare causal dilation convolution with fixed dilation rate convolution

Table 15 analyzes the difference between using causal dilation convolution (FUM(4)) and fixed dilation rate convolution (Fixed-d). The two methods use the same configuration, except for a difference in the size of the dilation rate. Causal dilation convolution using dilation rate described in the Eq. (9), while fixed dilation rate using a fixed dilation rate 2. It is found that FUM(4) achieves a more significant advantage on both *mini*Imagenet and *mini*Imagenet→CUB. This is because casual dilated convolution can extract all features of the channel vector sequence using fewer levels. In comparison, the fixed dilation rate convolution requires more levels. When Fixed-d and FUM(4) use the same layers, FUM(4) can utilize more information and obtain better accuracy.

4.9. The impact of pre-trained model on few-shot domain generalization tasks

We analyze the impact of the pre-trained model on the few-shot domain generalization task in Table 16. FUM(2,2,2)_fix use the same configuration as FUM(2,2,2), except that the backbone of FUM(2,2,2)_fix is pre-trained with the training set of *mini*Imagenet (64 classes) and the parameters of the pre-trained backbone is fixed in the meta-training process. The experimental results show that using the pre-trained backbone improves the prediction results in the 5-way 1-shot paradigm. However, there is a decrease in the 5-way 5-shot paradigm. We supposed that although the backbone network pre-trained with the full training set of *mini*Imagenet has good discriminative power, the training strategy biases the backbone network towards the classes of the *mini*Imagenet training set and makes it difficult to generalize to new domains.

5. Conclusion

In this paper, we move forward to handle the challenge of domain shifts in the context of FSL. This paper designed channel vector sequence containing relational information, which implies re-

lated information and helps infer the category of the query sample. The proposed forget-update module composed of stacked of forget-update blocks. The forgetting block retains useful information, and the updating block generates new features according to a specific scenario. The combination of channel vector sequence and forget-update module can generate relationship embeddings, which implies a similar relationship between a query sample and the support samples. Visualization experiments show that /fum can adjust the distribution of relational embedding across domains through forgetting and updating mechanisms. In the future, we will study the effectiveness of the FUM on Real-life scenarios.

Declaration of Competing Interest

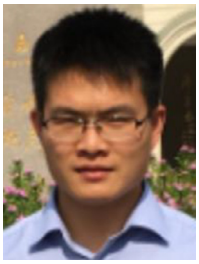
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by the Scientific Foundation of State Grid Corporation of China (Research on Ice-wind Disaster Feature Recognition and Prediction by Few-shot Machine Learning in Transmission Lines).

References

[1] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Y. Bengio, Y. LeCun (Eds.), ICLR, 2015.
[2] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016, pp. 770–778.
[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: CVPR, 2016, pp. 2818–2826.
[4] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: CVPR, 2018, pp. 7132–7141.
[5] B.M. Lake, R. Salakhutdinov, J.B. Tenenbaum, Human-level concept learning through probabilistic program induction, Science 350 (6266) (2015) 1332–1338.
[6] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, in: NIPS, 2016, pp. 3630–3638.
[7] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: ICML, 2017, pp. 1126–1135.
[8] J. Snell, K. Swersky, R.S. Zemel, Prototypical networks for few-shot learning, in: NIPS, 2017, pp. 4077–4087.
[9] F. Sung, Y. Yang, L. Zhang, T. Xiang, P.H.S. Torr, T.M. Hospedales, Learning to compare: relation network for few-shot learning, in: CVPR, 2018, pp. 1199–1208.
[10] H.-J. Ye, H. Hu, D.-C. Zhan, F. Sha, Few-shot learning via embedding adaptation with set-to-set functions, in: CVPR, 2020, pp. 8808–8817.
[11] R. Krishnan, S. Sarkar, Conditional distance based matching for one-shot gesture recognition, Pattern Recognit. 48 (4) (2015) 1302–1314.
[12] L. Zhang, X. Chang, J. Liu, M. Luo, M. Prakash, A.G. Hauptmann, Few-shot activity recognition with cross-modal memory network, Pattern Recognit. 108 (2020) 107348.
[13] A.K. Bhunia, A.K. Bhunia, S. Ghose, A. Das, P.P. Roy, U. Pal, A deep one-shot network for query-based logo retrieval, Pattern Recognit. 96 (2019).
[14] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C.F. Wang, J.-B. Huang, A closer look at few-shot classification, ICLR, 2019.
[15] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200-2011 dataset(2011).
[16] M.L. van der, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (11) (2008) 2579–2605.
[17] G. Koch, R. Zemel, R. Salakhutdinov, Siamese neural networks for one-shot image recognition, ICML Deep Learning Workshop, vol. 2, 2015.
[18] H. Qi, M. Brown, D.G. Lowe, Low-shot learning with imprinted weights, in: CVPR, 2018, pp. 5822–5830.
[19] S. Qiao, C. Liu, W. Shen, A.L. Yuille, Few-shot image recognition by predicting parameters from activations, in: CVPR, 2018, pp. 7229–7238.
[20] N. Mishra, M. Rohaninejad, X. Chen, P. Abbeel, A simple neural attentive meta-learner, ICLR, 2018.
[21] H. Huang, Z. Wu, W. Li, J. Huo, Y. Gao, Local descriptor-based multi-prototype network for few-shot learning, Pattern Recognit. 116 (2021) 107935.
[22] Z. Hu, Z. Li, X. Wang, S. Zheng, Unsupervised descriptor selection based meta-learning networks for few-shot classification, Pattern Recognit. 122 (2022) 108304.
[23] B. Zhang, X. Li, Y. Ye, Z. Huang, L. Zhang, Prototype completion with primitive knowledge for few-shot learning, in: CVPR, 2021, pp. 3754–3762.
[24] S. Qiao, C. Liu, W. Shen, A.L. Yuille, Few-shot image recognition by predicting parameters from activations, in: CVPR, 2018, pp. 7229–7238.
[25] A. Rozantsev, M. Salzmann, P. Fua, Beyond sharing weights for deep domain adaptation, IEEE TPAMI 41 (4) (2019) 801–814.
[26] B. Sun, J. Feng, K. Saenko, Return of frustratingly easy domain adaptation, in: D. Schuurmans, M.P. Wellman (Eds.), AAAI, 2016, pp. 2058–2065.
[27] M.-Y. Liu, O. Tuzel, Coupled generative adversarial networks, in: D.D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, R. Garnett (Eds.), NIPS, 2016, pp. 469–477.
[28] D. Yoo, N. Kim, S. Park, A.S. Paek, I.-S. Kweon, Pixel-level domain transfer, in: ECCV, vol. 9912, 2016, pp. 517–532.
[29] M. Ghifary, W.B. Kleijn, M. Zhang, D. Balduzzi, W. Li, Deep reconstruction–classification networks for unsupervised domain adaptation, in: ECCV, 2016, pp. 597–613.
[30] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: ICCV, 2017, pp. 2242–2251.
[31] M.-E. Nilsback, A. Zisserman, Automated flower classification over a large number of classes, in: Indian Conference on Computer Vision, Graphics and Image Processing, 2008.
[32] A. Khosla, N. Jayadevaprakash, B. Yao, L. Fei-Fei, Novel dataset for fine-grained image categorization, CVPR Workshop, Colorado Springs, CO, 2011.
[33] J. Krause, M. Stark, J. Deng, L. Fei-Fei, 3D object representations for fine-grained categorization, 3dRR-13, Sydney, Australia, 2013.
[34] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, B. Wang, Moment matching for multi-source domain adaptation, in: ICCV, 2019, pp. 1406–1415.
[35] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A.W. Senior, K. Kavukcuoglu, WaveNet: a generative model for raw audio, CoRR (2016) arXiv preprint arXiv:1609.03499.
[36] A. Van Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel recurrent neural networks, in: ICML, PMLR, 2016, pp. 1747–1756.
[37] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, ICLR, 2015.
[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, F.-F. Li, ImageNet: a large-scale hierarchical image database, in: CVPR, 2009, pp. 248–255.
[39] N. Hilliard, L. Phillips, S. Howland, A. Yankov, C.D. Corley, N.O. Hodas, Few-shot learning with metric-agnostic conditional embeddings, CoRR (2018) arXiv preprint arXiv:1802.04376.
[40] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: ICCV, 2015, pp. 1026–1034.
[41] T. Salimans, D.P. Kingma, Weight normalization: a simple reparameterization to accelerate training of deep neural networks, in: NIPS, 2016, p. 901.
[42] L. Zhang, J. Liu, M. Luo, X. Chang, Q. Zheng, A.G. Hauptmann, Scheduled sampling for one-shot learning via matching network, Pattern Recognit. 96 (2019).
[43] X. Li, L. Yu, C.-W. Fu, M. Fang, P.-A. Heng, Revisiting metric learning for few-shot image classification, Neurocomputing 406 (2020) 49–58.
[44] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, CoRR (2018) arXiv preprint arXiv:1803.01271.



Minglei Yuan is currently pursuing the PhD degree in the Department of Computer Science and Technology, Nanjing University. He received BS Degree from University of Science and Technology of China in 2012. His current interests are in the areas of few-shot learning, computer vision, and pattern recognition algorithms.

Chunhao Cai is currently pursuing the BS degree in the Department of Computer Science and Technology, Nanjing University. His current interests are in the areas of few-shot learning, and pattern recognition algorithms.

Tong Lu received the PhD degree in computer science from Nanjing University in 2005. He served as Associate Professor and Assistant Professor in the Department of Computer Science and Technology at Nanjing University from 2007 and 2005. He is now a full Professor at the same university. He also has served as Visiting Scholar at National University of Singapore and Department of Computer Science and Engineering, Hong Kong University of Science and Technology, respectively. He is also a member of the National Key Laboratory of Novel Software Technology in China. He has published over 130 papers and authored 2 books in his area of interest, and issued more than 20 international or Chinese invention patents. His current interests are in the areas of multimedia, computer vision and pattern recognition algorithms/systems. Dr. Tong Lu was a member of ACM, IAPR, ISAI and a senior member of China Computer Federation (CCF). He is the Youth Associate Editor of Journal on Frontiers of Computer Science (FCS), and has served as the Secretary-general of CAD&CG Committee of Jiangsu Computer Federation in China since 2008. He has been member of the program committee or session chair of more than 10 international scientific conferences, and the Chair of Organization Committee of Youth Scholar Forum of State Key Laboratory for Novel Software Technology since 2010.

Yirui Wu is currently an Associate Professor at Hohai University and working in Hydrology Big Data Group. Before coming to Hohai, I obtained my PhD degree from Nanjing University in 2016. I received my BS Degree from Nanjing University in 2011 as well. During my PhD study, I was with the IMAGE Lab under the supervision of Prof. Tong Lu and worked closely with Dr. Shivakumara Palaiahnakote.

Qian Xu is currently pursuing the BS degree in the Department of Computer Science and Technology, Nanjing University. Her current interests are in the areas of few-shot learning, and Image classification algorithms.

Shijie Zhou is the research leader of Jiangsu Welm Technology. He received his MSc and BSc from University of Sydney, and Jiangsu University of Science and Technology, respectively. His current interests are in the areas of pattern recognition and artificial intelligence applications.