Few-shot Unsupervised Domain Adaptation with Image-to-class Sparse Similarity Encoding

Shengqi Huang  
Nanjing Normal University  
Nanjing, China  
huangshengqi@njnu.edu.cn

Wanqi Yang  
Nanjing Normal University  
Nanjing, China  
yangwq@njnu.edu.cn

Lei Wang  
University of Wollongong  
Wollongong, Australia  
leiw@uow.edu.au

Luping Zhou  
University of Sydney  
Sydney, Australia  
luping.zhou@sydney.edu.au

Ming Yang  
Nanjing Normal University  
Nanjing, China  
myang@njnu.edu.cn

ABSTRACT

This paper investigates a valuable setting called few-shot unsupervised domain adaptation (FS-UDA), which has not been sufficiently studied in the literature. In this setting, the source domain data are labelled, but with few-shot per category, while the target domain data are unlabelled. To address the FS-UDA setting, we develop a general UDA model to solve the following two key issues: the few-shot labeled data per category and the domain adaptation between support and query sets. Our model is general in that once trained it will be able to be applied to various FS-UDA tasks from the same source and target domains. Inspired by the recent local descriptor based few-shot learning (FSL), our general UDA model is fully built upon local descriptors (LDs) for image classification and domain adaptation. By proposing a novel concept called similarity patterns (SPs), our model not only effectively considers the spatial relationship of LDs that was ignored in previous FSL methods, but also makes the learned image similarity better serve the required domain alignment. Specifically, we propose a novel Image-to-class sparse Similarity Encoding (IMSE) method. It learns SPs to extract the local discriminative information for classification and meanwhile aligns the covariance matrix of the SPs for domain adaptation. Also, domain adversarial training and multi-scale local feature matching are performed upon LDs. Extensive experiments conducted on a multi-domain benchmark dataset DomainNet demonstrates the state-of-the-art performance of our IMSE for the novel setting of FS-UDA. In addition, for FSL, our IMSE can also show better performance than most of recent FSL methods on miniImageNet.

CCS CONCEPTS

• Computing methodologies → Transfer learning: Learning settings.

Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MM ’21, October 20–24, 2021, Virtual Event, China  
© 2021 Association for Computing Machinery.  
ACM ISBN 978-1-4503-8651-7/21/10 . . $15.00  
https://doi.org/10.1145/3474085.3475232

1 INTRODUCTION

Unsupervised domain adaptation (UDA) leverages sufficient labeled data in a source domain to classify unlabeled data in a target domain [23, 29, 30]. Since the source domain usually has a different distribution from the target domain, most existing UDA methods [3, 10, 23] have been developed to reduce the domain gap between both domains. However, in real cases, when the source domain suffers from high annotation cost or the limited access to labeled data, the labeled data in source domain could become seriously insufficient. This makes the learning of a UDA model more challenging. On the other hand, existing UDA methods usually train a specific UDA model for a certain task with given categories, and thus cannot be efficiently updated to classify unseen categories in a new task. Recently, the need of handling few-shot labeled source domain data and having a general UDA model that can quickly adapt to many new tasks has been seen in a number of real applications.

For example, in the online shopping, sellers generally show a few photos professionally taken by photographers for each product (as source domain), while buyers usually search for their expected products by using the photos directly taken by their cellphones (as target domain). Since different buyers usually submit the images of products with different categories and types, the online shopping system needs to have a general domain adaptation model to classify the types of these products, whatever their categories are. Also, we take another example. The database of ID card usually only stores one photo for each person and these ID photos are captured in photo studios with well controlled in-door lighting conditions (as source domain). However, in many cases, an intelligence system needs to identity these persons by matching with their facial pictures captured by smartphones, ATM cameras, or even surveillance cameras in open environments (as target domain).
This situation motivates us to focus on a valuable but insufficiently studied setting, few-shot unsupervised domain adaptation (FS-UDA). A FS-UDA task includes a support set from the source domain and a query set from the target domain. The support set contains few labeled source domain samples per category, while the query set contains unlabeled target domain samples to be classified. The goal is to train a general model to handle various FS-UDA tasks from the same source and target domains. These tasks realize classification of different category sets in the target domain.

Our solution to this FS-UDA setting is inspired by few-shot learning (FSL). Recent metric based FSL methods [13, 24, 31, 35] usually learn a feature embedding model for image-to-class measurement, and meanwhile perform episodic training to make the model generalized to new tasks without any retraining or fine-tuning process. Considering the domain gap between the support and query sets in the proposed setting of FS-UDA, we tactically perform episodic training on an auxiliary dataset that contains the data from both domains, aiming to learn a general and domain-shared feature embedding model. For each episode, a few samples from both domains are randomly sampled from the auxiliary dataset to train the model.

Also, for efficient image-to-class measurement, several recent methods [12, 13, 35] leveraged local descriptors (LDs) of images, because they can better provide discriminative and transferable information across categories, which can be important cues for image classification in FSL [35]. These methods first measured the similarities (or distances) between the LDs of a query image and a support class by using cosine [13], KL divergence [12] or EMD distance [35], and then combined these similarities to produce a single image-to-class similarity score for classification. We believe that for two LDs that are spatially adjacent and therefore highly likely similar to each other, their similarity values to other LDs should be close and change in a smooth manner. However, the above FSL methods did not effectively take the spatial relationship of the LDs into account, and could result in noisy or non-smooth similarities w.r.t. the spatial dimension (e.g. caused by outliers, occlusion or background clutter in images). Moreover, considering the key role of image-to-class similarities for classification, we leverage them to align the source and target domains. Nevertheless, merely having a single similarity score as produced in the above methods cannot effectively serve the domain alignment required in our FS-UDA.

Therefore, instead of computing a single similarity score, we design a novel image-to-class similarity pattern (SP) vector that contains the similarities of a query image to all LDs of a support class. We design a Gaussian filter and a max-pooling module on the spatial dimension of query images, which can smooth the similarities of spatially-adjacent LDs and extract the prominent similarity. Thus, SP provides a fine-grained and patterned measurement for classification. Also, the SPs of query images can be leveraged as their features for domain alignment. To this end, we align the covariance matrix of the SPs between domains, because covariance matrix can naturally capture the underlying data distribution [14]. For further domain alignment, we also perform domain adversarial training and multi-scale matching on LDs. In doing so, we propose a novel method, Image-to-class sparse Similarity Encoding (IMSE), which learns image-to-class similarity patterns to realize effective classification and domain adaptation. Figure 1 illustrates the encoding process of SPs. Our contributions can be summarized as:

1. **A novel method for the new setting of FS-UDA.** We propose a novel method IMSE for a valuable but insufficiently studied setting FS-UDA, which learns a domain-shared feature embedding model and adapts this general model to handle various new tasks.

2. **A novel image-to-class measurement between source and target domains.** We propose image-to-class similarity patterns that consider the spatial information of local descriptors. Moreover, we leverage the similarity patterns and local descriptors to further align the domains.

3. **Effective performance.** Extensive experiments are conducted on a multi-domain benchmark dataset DomainNet, showing the state-of-the-art performance of our method for FS-UDA. In addition, for FSL, our IMSE also shows better performance than most recent FSL methods on miniImageNet.

## 2 RELATED WORK

**Unsupervised domain adaptation.** The UDA setting aims to reduce domain gap and leverage sufficient labeled source domain data to realize classification in target domain. Many UDA methods [16, 30] are based on maximum mean discrepancy to minimize the feature difference across domains. Besides, adversarial training is widely used [7, 29] to tackle domain shift. ADDA [29] performs domain adaptation by using an adversarial loss and learning...
two domain-specific encoders. The maximum classifier discrepancy method (MCD) [23] applies task-specific classifiers to perform adversarial learning with the feature generator. Besides, several methods [3, 10, 32] train the classifier in target domain by leveraging their pseudo-labels. SDRC [26] learns clustering assignment of target domain and leverage them as the pseudo-labels to compute classification loss. In sum, these UDA methods require sufficient labeled source domain data for domain alignment and classification, but would not work when encountering the issues of scarce labeled source domain and task-level generalization that exist in FS-UDA.

**Few-shot learning.** Few-shot learning has become one of the most popular research topics recently, which aims to quickly adapt the model to new few-shot tasks. It contains two main branches: optimization-based and metric-based approaches. Optimization-based methods [2, 6, 21] leverage a meta learner over the auxiliary dataset to learn a general initialization model. When handling new tasks, the initialization model can be quickly fine-tuned to adapt to new tasks. Metric-based methods [13, 24, 31, 34] aim to learn a generalizable feature embedding metric space by episodic training on the auxiliary dataset [31]. This metric space can be directly applied to new few-shot tasks without any fine-tuning or re-training. Our method is related to metric-based methods. Classically, ProtoNet [24] learns the class prototypes in the support set and classifies the *query images* by calculating the maximum similarity to these prototypes. CAN [9] learns the cross-attention map between *query images* and *support classes* to highlight the discriminative regions. Due to the local invariance of local descriptors (LDs), several methods [12, 13, 15, 35] leverage large amounts of LDs to solve the FSL problem. For example, DN4 [13] classifies *query images* by finding the neighborhoods of their LDs amongs LDs of the *support classes*. Based on LDs, DeepEMD [35] measures earth mover’s distance of *query images* to the structured prototype of *support classes* for *image-to-class* optimal matching. For performance improvement, recent FSL methods usually pre-train the embedding network [5] on auxiliary dataset. Inspired by these methods, we leverage LDs to calculate *image-to-class* similarity patterns (SPs) for classification and domain alignment in our setting. Furthermore, a few recent works focus on the issue of *cross-domain FSL* in which domain shift exists between data of meta tasks and new tasks. Chen et al. [4] leverage a baseline model to do cross-domain FSL. LFT [28] designs a feature transformation strategy to tackle the domain shift.

**Remarks.** Note that the setting of FS-UDA is crucially different from the existing settings: UDA [23, 29], FSL [13, 35] and cross-domain FSL [4, 28]. As shown in Table 1, compared with UDA, FS-UDA is to deal with many UDA tasks by leveraging few labeled source domain samples for each. As known, FSL is to train a general classifier by leveraging few-shot training samples per category and adapt the classifier to new tasks. The new tasks for FSL are sampled from the same domain as the tasks for training, while the new tasks for cross-domain FSL could be from new domains. Compared with FS-UDA, FSL and cross-domain FSL are not capable of handling the issues of no available labels in the target domain, and the domain gap between the support and query sets in every task, while these issues will be solved in our FS-UDA. In addition, the FS-UDA setting also has key differences from existing few-shot domain adaptation methods [18, 27, 33], because i) the “few-shot” in the latter is referred to labeled target domain data and ii) they still need sufficient labeled source domain data to work. For the one-shot UDA recently proposed in [17], it deals with the case that only one unlabeled target sample is available, but does not require the source domain to be few-shot, which is different from ours. In short, the above methods focus on data or label scarcity in target domain rather than in source domain. Also, they do not consider developing a single model that can generally be adapted to many different UDA tasks. These differences indicate the necessity of the novel FS-UDA setting, for which we will explore a new learning algorithm to address the data scarcity in source domain and the UDA-task-level generalization.

### 3 METHODOLOGY

#### 3.1 Problem Definition

**A N-way, K-shot UDA task.** Our FS-UDA setting involves two domains in total: a source domain $S$ and a target domain $T$. A *N-way, K-shot UDA task* includes a support set $X_S^T$ from $S$ and a query set $Q_T$ from $T$. The support set $X_S^T$ contains $N$ classes and $K$ source domain samples per class. The query set $Q_T$ contains the same $N$ classes as in $X_S$ and $N_q$ target domain samples per class. To classify *query images* in $Q_T$ to the correct class in $X_S$, as a conventional solution, leveraging the insufficient support set $X_S$ to train a specific model from scratch could make the classification inefficient. Therefore, we aim to train a general model that can handle many new N-way, K-shot UDA tasks for testing. Table 2 shows the main symbols used in this paper.

**Auxiliary dataset and episodic training.** Inspired by FSL, we resort to an auxiliary dataset $D_{aux}$ and perform episodic training to learn the general model. The auxiliary dataset $D_{aux}$ includes labeled source domain data and unlabeled target domain data. Note that, the classes in $D_{aux}$ are completely different from the testing N-way, K-shot UDA tasks. That is, the classes of the testing UDA tasks are unseen during episodic training. We construct large amounts of episodes to simulate the testing UDA tasks for task-level generalization [31]. Each episode contains a support set $X_S$, a source domain query set $Q_S$ and a target domain query set $Q_T$ sampled from $D_{aux}$. Specifically, in each episode, we randomly sample $N$ classes and $K + N_q$ source domain samples per class from $D_{aux}$. The $K + N_q$ samples per class are partitioned into $K$ in $X_S$ and the remaining $N_q$ in $Q_S$. Meanwhile, we randomly sample $N + N_q$ target domain samples (without considering labels) from $D_{aux}$ and incorporate

| Settings                      | Domain gap between the support and query sets | No available labels in the target domain | Few-shot labeled samples per category | Handling many tasks with different sets of categories |
|-------------------------------|-----------------------------------------------|----------------------------------------|--------------------------------------|-----------------------------------------------------|
| UDA                           | ✓                                              | ✓                                      | ✓                                    | ✓                                                   |
| FSL                           | ✓                                              | ✓                                      | ✓                                    | ✓                                                   |
| Cross-domain FSL              | ✓                                              | ✓                                      | ✓                                    | ✓                                                   |
| FS-UDA (ours)                 | ✓                                              | ✓                                      | ✓                                    | ✓(tasks from new domains)                            |
Figure 2: Illustration of our method performing episodic training for 5-way, 1-shot UDA tasks. In each episode, the support set \( X_S \) contains a photo image per support class, and the query sets (\( Q_S \) and \( Q_T \)) contains the photo and sketch query images, respectively. Support class \( i \) and query images \( q_s, q_t \) from both domains are first through the feature embedding network \( F \) to extract their local descriptors (LDs), followed by similarity pattern encoding module to learn similarity patterns (SPs). Then, we leverage the SPs in \( Q_S \) to calculate the similarity of query images to the class \( i \) for classification loss \( L_{cls} \). Meanwhile, we calculate the covariance matrix of the SPs between \( Q_S \) and \( Q_T \) for measuring their domain alignment loss \( L_{spa} \). In addition, the LDs from both domains are further aligned by calculating both adversarial training loss \( L_{adv} \) and multi-scale matching loss \( L_{msm} \). Finally, the above losses are backpropagated to update the embedding model \( F \).

### 3.2 Similarity Pattern Encoding

Inspired by LD-based FSL methods [12, 13, 35], we employ plenty of local descriptors (LDs) to measure the image-to-class similarity. Considering spatial relationship of LDs, we believe that for two LDs that are spatially adjacent and therefore highly likely similar to each other, their similarity values to other LDs should be close and change in a smooth manner. However, the above methods did not effectively take this into account, and could generate noisy or non-smooth similarities with respect to the spatial dimension. Moreover, merely having a single similarity score as in the above methods cannot effectively serve the domain alignment required in our FS-UDA. Therefore, we propose a similarity encoding module to learn image-to-class similarity patterns (SPs), by leveraging Gaussian filter and max-pooling to smooth and aggregate the similarities of LDs. The learning process of SPs is cast into three following parts:

1. **Computing the similarity matrix of LDs (see step 1 in Figure 1)** as in DN4 [13]. We firstly extract the LDs of a query image \( q \) and a support class \( c (c = 1, \ldots, N) \), and then construct their LD matrix \( L_{q}^{el} \in \mathbb{R}^{HW \times C} \) and \( L_{sup}^{el} \in \mathbb{R}^{KHW \times C} \), respectively, where each row represents a LD vector. \( H, W \) and \( C \) denote the height, the width and the number of channels of the feature map.

### Table 2: Notations.

| Notations | Descriptions |
|-----------|--------------|
| \( N, K \in \mathbb{R} \) | The number of the classes and samples per class |
| \( \lambda_s, \lambda_a, \lambda_m \in \mathbb{R} \) | Weight parameters of three loss terms in Eq. (5) |
| \( X_S, Q_S, Q_T \) | Support set of source domain, and query sets of source domain and target domain |
| \( H, W, C \in \mathbb{R} \) | The height, the width and the number of channels, and local descriptor (LD) vector in the feature map |
| \( L_{q}^{el} \in \mathbb{R}^{HW \times C} \) | LD matrix that consists of LDs of a query image \( q \) |
| \( L_{sup}^{el} \in \mathbb{R}^{KHW \times C} \) | LD matrix that consists of LDs of a support class \( c \) |
| \( M_{q}^{el} \in \mathbb{R}^{HW \times HW} \) | 2D and 3D matrices of cosine similarities between LDs of a query image \( q \) and a support class \( c \) |
| \( M_{sup}^{el} \in \mathbb{R}^{KHW \times KHW} \) | 2D and 3D matrices of cosine similarities between LDs of a support class \( c \) and a support class \( c' \) |
| \( S_{q}^{el} \in \mathbb{R}^{HW \times HW} \) | Covariance matrix of similarity patterns from all query images in domain \( d \) to a support image \( i \) |
| \( F, D \) | Feature embedding network, domain discriminator |

them into \( Q_T \). Note that the query set \( Q_S \) per episode is additional, compared with the testing tasks. This is because the labels of \( Q_T \) are unavailable, we instead leverage \( Q_S \) to calculate classification loss, and meanwhile use \( Q_S \) and \( Q_T \) for domain alignment.

**The flowchart of our method.** Figure 2 illustrates our method to perform episodic training for 5-way, 1-shot UDA tasks. In each episode, a support set \( (X_S) \) and two query sets \( (Q_S, Q_T) \) are first through the feature embedding network \( F \) to extract their LDs, followed by similarity encoding module to learn similarity patterns (SPs). Then, we leverage the SPs to calculate the image-to-class similarity for classification loss \( L_{cls} \). Meanwhile, we calculate the covariance matrix of the SPs between domains to measure their domain alignment loss \( L_{spa} \). In addition, the LDs from domains are further aligned by calculating both adversarial training loss \( L_{adv} \) and multi-scale matching loss \( L_{msm} \). Finally, the above losses are backpropagated to update the embedding model \( F \).
respectively. Then, we compute their cosine similarity matrix $M_q^C \in \mathbb{R}^{HW \times HW'}$ by $\cos(L_q^{\text{que}}, L_q^{\text{supp}})$. Finally, we only maintain the top-K similarity values for each row in $M_q^C$ and set the remaining as zero.

(2) Reshaping the similarity matrix $M_q^C$ into a 3D map (see step 2 in Figure 1). To recover the spatial positions of query images in the similarity matrix, we reshape the similarity matrix $M_q^C$ into a 3D similarity representation $M_{q, \text{spa}} \in \mathbb{R}^{HW \times HW \times W}$, by partitioning its first dimension $HW$ to the two dimensions $H$ and $W$.

(3) Performing Gaussian filter and max-pooling to generate the SPs (see step 3 in Figure 1). According to our observation, several noises or non-smooth values exist on the first two-dimensional plane of $M_q^C$. Considering that similarities of spatially-adjacent LDs to other LDs should be close, we conduct Gaussian filtering (with $3 \times 3 \times 1$ size) from the first two dimensions of $M_q^C$ to smooth their similarities. Then, max-pooling is employed to extract the discriminative similarity and finally generate the SP vectors ${p_q^C}$. Compared with a single similarity score used in [13], a SP contains the similarities of a query image to all LDs of the support class. Thus, the SP provides an informative and patterned measurement, which will be used for classification and domain alignment.

For classification, we calculate the image-to-class similarity score for each support class by summing all the elements of the SP vector, i.e. $\exp(1^T \cdot p_q^C)$, and then select the class label with the highest score as the predicted label of the query image $q$. The classification loss $L_{\text{cls}}$ is calculated by cross-entropy loss, which can be written by:

$$L_{\text{cls}} = -\frac{1}{|Q_S|} \sum_{q \in Q_S} \log \exp(1^T \cdot p_q^C),$$

where $c^*$ is the actual label of the query image $q$.

### 3.3 Similarity Pattern Alignment

To reduce the domain gap, we align the SPs of query images from the source and target domains, because the SPs contain image-to-class similarities that are crucial to final classification predictions. Considering that covariance matrix could exactly capture the underlying distribution information [1, 8, 14], we align the covariance matrix of the SPs from both domains for domain adaptation.

For fine-grained alignment, we focus on the SP between a query image $q$ and a specific support image $i$ (namely image-to-image SP, $p_q^i \in \mathbb{R}^{HW}$), instead of the image-to-class one. In other words, the learned image-to-image pattern $p_q^i \in \mathbb{R}^{HW}$ can be decomposed into $K$ different image-to-image patterns $p_q^i \in \mathbb{R}^{HW \times 1}$, $i = 1, ..., K$.

Specifically, for a support image $i$, we first collect its image-to-image SPs $E_q^i \in \mathbb{R}^{Q_S \times 1}$, i.e. $E_q^i = \{p_q^i | q \in Q_S\}$. Then, considering that covariance matrix $\Sigma_q^i \in \mathbb{R}^{HW \times HW}$ for both domains $d \in (S, T)$ can be calculated by

$$\Sigma_q^i = \frac{1}{|E_q^i|-1} \sum_{p_q^i \in E_q^i} (p_q^i - \bar{p}^i)(p_q^i - \bar{p}^i)^T,$$

where $\bar{p}^i$ represents the averaged vector of SPs in the set $E_q^i$. Finally, to align the two domains, we design a loss function of similarity pattern alignment (spa) to minimize the difference between $\Sigma_q^S$ and $\Sigma_q^T$, which can be calculated as:

$$L_{\text{spa}}({\Sigma}_q^S, {\Sigma}_q^T) = \frac{1}{NK} \sum_{i=1}^{NK} ||\Sigma_q^S - \Sigma_q^T||_F^2,$$

where $NK$ represents the number of images in $N$-way, $K$-shot support set. By minimizing the loss $L_{\text{spa}}$, the domain gap will be reduced. In some sense, this loss can also be viewed as a regularization of image-to-class similarity measurement. This will be also validated in the experiments of few-shot learning.

### 3.4 Local Descriptor Alignment

**Domain Adversarial Module.** To align the domains on local features, we apply adversarial training on the LDs of query images between $Q_S$ and $Q_T$, which are from the distributions $P_S$ and $P_T$, respectively. We design a domain discriminator $D$ to predict the domain label of each LD $l \in \mathbb{R}^{C}$. The adversarial training loss can be written by:

$$\min_{F} \max_{D} L_{\text{adv}} = E_{l \sim P_S} \log(D(l)) + E_{l \sim P_T} \log(1 - D(l)).$$

In each episode, we minimize the loss $L_{\text{adv}}$ for the embedding network $F$ and maximize the loss for the discriminator $D$.

**Multi-scale Local Descriptor Matching.** In addition, to incorporate the discriminative information of the target domain for domain alignment, we develop a multi-scale local descriptor matching (msm) module between domains. It is encouraged to push each LD $l_i$ in $Q_T$ to its $k$-th nearest neighbors from multi-scale LDs in $Q_S$ in an unsupervised way. These multi-scale LDs are generated by average pooling on embedded feature maps with different scales ($5 \times 5, 2 \times 2, 1 \times 1$), and cosine similarity is utilized to find the nearest neighbors. This loss can be written as:

$$L_{\text{msm}} = \frac{1}{|Q_T|} \sum_{q \in Q_T} \sum_{l \in Q_S} \sum_{i=1}^{k} \log \frac{\exp(m_{ij}^l)}{\sum_{j=1}^{c} \exp(m_{ij}^l)},$$

where $m_{ij}^l \in [-1, 1]$ represents the cosine similarity of the LD $l_i$ to its $j$-th neighbor in multi-scale LDs of $Q_S$. Also, for every query image, we only measure the similarity of its $n$ neighbors, and the remaining neighbors are not considered.

In sum, we combine the above losses to train the feature embedding network $F$ on a large amount of episodes. The overall objective function can be written as:

$$\min_{F} \max_{D} \left[L_{\text{cls}} + \lambda_{\text{spa}} L_{\text{spa}} + \lambda_{\text{adv}} L_{\text{adv}} + \lambda_{\text{msm}} L_{\text{msm}}, \right],$$

where the parameters $\lambda_{\text{spa}}, \lambda_{\text{adv}}$, and $\lambda_{\text{msm}}$ are to balance the effects of different loss terms $L_{\text{spa}}, L_{\text{adv}}$, and $L_{\text{msm}}$, respectively. The whole process of the proposed IMSE is summarized in Algorithm 1.

### 4 EXPERIMENTS

**DomainNet dataset.** To demonstrate the efficacy of our method, we conduct extensive experiments on a multi-domain benchmark dataset DomainNet. It was released in 2019 for the research of multi-source domain adaptation [20]. It contains 345 categories and six domains per category, i.e. quickdraw, clipart, real, sketch, paint, and infograph domains. In our experiments, we first remove the data-insufficient domain infograph, and select any two from the remaining five domains as the source and target domains. Thus, we have 20 combinations totally for evaluation. Then, we discard the 37 categories that contain less than 20 images for the five domains. Finally, we randomly split the images of the remaining 308 categories into the images of 217 categories, 43 categories and 48 categories for episodic training (forming the auxiliary dataset), model validation.
Algorithm 1 Proposed IMSE
Training Input: An auxiliary dataset $D^{aux}$ including labeled source and unlabeled target domain data, parameters $\lambda_s, \lambda_d, \lambda_m$
Training Output: Embedding network $F$ and discriminator $D$
1: Randomly initialize $F$ and $D$.
2: while not converged do
3: Sample an episode $\{(X_S, Q_S, Q_T)\}$ from $D^{aux}$.
4: # compute similarity patterns
5: Calculate similarity patterns $\{p_{S}^{T}q|q \in Q_S \cup Q_T, c = 1, ..., N\}$ using $F(X_S), F(Q_S)$ and $F(Q_T)$.
6: # compute classification loss and domain alignment losses
7: Compute classification loss $L_{cls}$ for labeled query set $Q_S$.
8: Compute SP alignment loss $L_{spa}$ via Eq. (2).
9: Compute LD alignment losses $L_{adv}, L_{msm}$ via Eqs. (3)-(4).
10: # update parameters
11: Optimize the $F$ and $D$ by minimizing the objective in Eq. (5).
12: end while
Testing Input: New tasks and trained embedding network $F$
Testing Output: Predictions $\{\hat{y}_q\}$ in target domain of new tasks
1: for $\{(X_S, Q_T)\}$ in new tasks do
2: # test on a new task
3: Compute similarity patterns $\{p_{S}^{T}q|q \in Q_T, c = 1, ..., N\}$ using $F(X_S)$ and $F(Q_T)$.
4: Predict the labels $\{\hat{y}_q\}$, $\hat{y}_q = \arg \max_c \{1^T \cdot p_{S}^{T}q|c = 1, ..., N\}$.
5: end for

and testing new tasks, respectively. Note that in each split every category contains the images of the five domains.

Network architecture. We employ ResNet-12 as the backbone of the embedding network, which is widely used in few-shot learning [5]. As for similarity pattern encoder, we set a $3 \times 3 \times 1$ Gaussian kernel with standard deviation 0.8 and stride 1. Then, a $2 \times 2$ maxpooling with stride 2 is conducted. As for adversarial training, we use three fully-connected layers to build a domain discriminator.

Model training. Before episodic training, inspired by [5], a pre-trained embedding network is trained by source domain data in the auxiliary dataset, which is used for performance improvement. Specifically, we first train the ResNet-12 with a full-connected layer to classify the source domain images in the auxiliary dataset. After the pre-training, we remove the full-connected layer and retain the convolutional blocks of ResNet-12 as the initialized model of the episodic training. Afterwards, we perform episodic training on 10000 episodes. For each episode, we randomly sample a support set (including 5 categories, and 1 or 5 source domain samples per category) and a query set (including 15 source domain samples for each of the same 5 categories and 75 randomly sampled target domain samples). During episode training, the total loss in Eq. (5) is minimized to optimize the network parameters for each episode. Also, we employ Adam optimizer with an initial learning rate of $1e^{-4}$, and reduce the learning rate by half every 1000 episodes.

Model validation and testing. For model validation, we compare the performance of different model parameters on 3000 tasks, which is randomly sampled from the validate set containing 43 categories. Then, we select the model parameters with the best validation accuracy for testing. During the testing phase, we randomly construct 3000 new FS-UDA tasks from the testing set containing 48 categories. For each task, the support set includes 5 categories, and 1 or 5 source domain samples per category, and the query set only includes 75 target domain samples from the same 5 categories for evaluation. The averaged top-1 accuracy on the 3000 tasks is calculated as the evaluation criterion.

4.1 Comparison Experiments for FS-UDA
We conduct extensive experiments on DomainNet to compare our method with related FSL, UDA, and their combination methods.

UDA methods. We choose MCD [23], ADDA [29] and DWT [22] to compare with our method. Specifically, 1) we first train the UDA models using all training categories in the auxiliary set; 2) Secondly, we replace the last classification layer of the models with a new 5-way classifier for various tasks, as in baseline++ [4]; 3) Thirdly, for testing on each task, we freeze the trained backbone and only fine-tune the classifier for a novel task with its support set, by using an SGD optimizer (learning rate 0.01, momentum 0.9 and weight decay $1e^{-4}$) with batch size 4 for 300 iterations. Finally, we test the classification performance on its query set.

FSL methods. Since our method is related to metric-based FSL methods, we choose five representative metric-based methods, i.e. ProtoNet [24], DN4 [13], ADM [12], FEAT [34], and DeepEMD [35] for comparisons. Especially, DN4, ADM and DeepEMD also utilize local descriptors for classification. For fair comparison, we also pre-train the embedding network and perform episodic training on auxiliary dataset [5]. Note that, FSL methods don’t handle the domain shift between the support and query sets, and are also not capable of leveraging the target domain data in the auxiliary dataset because of no labels in the target domain. Thus, we further combine the FSL and UDA methods for domain adaptation and classification.

FSL methods combined with ADDA [29]. We combine the five above FSL methods with ADDA [29], which are abbreviated as ADDA+ProtoNet, ADDA+DN4, ADDA+ADM, ADDA+FEAT, and ADDA+DeepEMD, respectively. Specifically, we add the feature-level adversarial loss in ADDA [29] for domain alignment in every episode. The used discriminator is the same with our method. For fair comparison, these combination methods also pretrain the embedding network before episodic training.

Comparison Analysis. Table 3 shows the results of all the compared methods for 20 cross-domain combinations, which records the averaged classification accuracy of target domain samples over 3000 5-way 1-shot/5-shot UDA tasks. As reported in Table 3, our IMSE achieves the best performance for the most combinations and their average. Specifically, the UDA and FSL baselines in the first two parts of Table 3 perform much worse than our method. This is because the UDA models did not consider task-level generalization, and the FSL models are lack of handling domain shift. In the third part of Table 3, the combination methods we build perform domain adversarial training over every episode. They are thus generally better than the above two parts, but still inferior to our IMSE. This is because the combination methods with ADDA [29] only perform the domain alignment based on feature maps, not considering the alignment of both local descriptors (more fine-grained than feature
| Table 3: Comparison of our methods with the UDA, FSL, and their combination methods for 5-way-1-shot or 5-shot FS-UDA tasks. Each row represents the accuracy (%) of a compared method adapting between two domains, where the \( \text{skt}, \text{rel}, \text{qdr}, \text{pnt}, \text{cli}, \text{rel}, \text{qdr}, \text{pnt}, \text{cli} \) denote the sketch, real, quickdraw, painting, and clipart domains in the \( \text{DomainNet} \) dataset, respectively. The best results are in bold.

| Method          | 5-way-1-shot | 5-shot          |
|----------------|-------------|----------------|
| MCD [23]       | 48.07/37.74 | 49.81/38.89    |
| ADDA [29]      | 48.82/46.66 | 49.95/42.29    |
| DWT [22]       | 49.43/38.67 | 52.23/40.78    |
| ProtoNet [24]  | 48.05/43.15 | 51.95/45.32    |
| DN4 [13]       | 52.42/47.29 | 54.25/48.88    |
| ADN [12]       | 49.66/42.27 | 49.31/45.64    |
| FEAT [34]      | 51.72/46.66 | 46.22/49.92    |
| DeepEMD [35]   | 52.24/46.48 | 41.23/46.37    |
| ADDA+ProtoNet  | 50.13/43.47 | 50.62/43.91    |
| ADDA+DN4       | 53.04/46.68 | 46.23/46.68    |
| ADDA+ADN       | 51.87/43.09 | 49.18/45.14    |
| ADDA+DN4       | 57.22/46.80 | 47.00/36.94    |
| ADDA+DeepEMD   | 53.98/47.55 | 44.64/36.19    |

| IMSE (ours)    | 57.21/51.30 | 57.94/51.86    |

| Table 4: Abundance study of Gaussian kernel and max-pooling used in our IMSE to generate SPs.

| Filters | Poolings | Gaussian MaxPooling | Gaussian DownSampling | Average MaxPooling | None MaxPooling |
|---------|----------|----------------------|-----------------------|--------------------|-----------------|
| skt-rel | 57.21/51.30 | 49.71/40.91 | 50.36/43.65 | 46.44/36.35 | 59.34/45.06 |
| skt-qdr | 57.21/51.30 | 54.41/44.14 | 50.36/43.65 | 46.44/36.35 | 59.34/45.06 |
| skt-pnt | 57.21/51.30 | 55.97/46.66 | 50.36/43.65 | 46.44/36.35 | 59.34/45.06 |
| rel-rel | 57.21/51.30 | 51.95/45.32 | 49.31/45.64 | 49.31/45.64 | 49.31/45.64 |
| rel-qdr | 57.21/51.30 | 49.95/42.29 | 49.31/45.64 | 49.31/45.64 | 49.31/45.64 |
| rel-pnt | 57.21/51.30 | 49.95/42.29 | 49.31/45.64 | 49.31/45.64 | 49.31/45.64 |
| cli-cll | 57.21/51.30 | 49.95/42.29 | 49.31/45.64 | 49.31/45.64 | 49.31/45.64 |
| qdr-qdr | 57.21/51.30 | 49.95/42.29 | 49.31/45.64 | 49.31/45.64 | 49.31/45.64 |
| qdr-pnt | 57.21/51.30 | 49.95/42.29 | 49.31/45.64 | 49.31/45.64 | 49.31/45.64 |
| pnt-pnt | 57.21/51.30 | 49.95/42.29 | 49.31/45.64 | 49.31/45.64 | 49.31/45.64 |

| Filters | Poolings | Gaussian MaxPooling | Gaussian DownSampling | Average MaxPooling | None MaxPooling |
|---------|----------|----------------------|-----------------------|--------------------|-----------------|
| skt-rel | 48.82/46.66 | 38.90/34.51 | 39.33/35.59 | 39.33/35.59 | 39.33/35.59 |
| skt-qdr | 48.82/46.66 | 50.67/47.16 | 31.78/35.47 | 31.78/35.47 | 31.78/35.47 |
| skt-pnt | 48.82/46.66 | 45.81/40.50 | 28.92/29.99 | 28.92/29.99 | 28.92/29.99 |
| rel-rel | 48.82/46.66 | 43.91/32.38 | 43.91/32.38 | 43.91/32.38 | 43.91/32.38 |
| rel-qdr | 48.82/46.66 | 47.86/37.74 | 38.90/34.51 | 38.90/34.51 | 38.90/34.51 |
| rel-pnt | 48.82/46.66 | 46.86/37.74 | 38.90/34.51 | 38.90/34.51 | 38.90/34.51 |
| cli-cll | 48.82/46.66 | 39.31/35.59 | 51.43/38.98 | 51.43/38.98 | 51.43/38.98 |
| qdr-qdr | 48.82/46.66 | 54.52/51.55 | 54.52/51.55 | 54.52/51.55 | 54.52/51.55 |
| qdr-pnt | 48.82/46.66 | 45.81/40.50 | 28.92/29.99 | 28.92/29.99 | 28.92/29.99 |
| pnt-pnt | 48.82/46.66 | 45.81/40.50 | 28.92/29.99 | 28.92/29.99 | 28.92/29.99 |

**maps** and similarity patterns (related to classification predictions) used in our IMSE. This shows the efficacy of our IMSE for FS-UDA tasks. On the other hand, we can see that the 20 cross-domain combinations have considerably different performances. This is because several domains (e.g., quickdraw) are significantly different from other domains, while some other domains (e.g., real, painting) are with the similar styles and features. Thus, the performance becomes relatively low when the domain gap is presented, and the situation will become better when the domain reduces. For example, from quickdraw to painting/real, most methods perform worse due to larger domain gap, but our IMSE and ADDA [29] perform much better than the others, showing the positive effect of their domain alignment. From painting to real and real to painting which have smaller domain gap, the FSL, combination methods and our IMSE perform better than the UDA methods, showing the positive effect of task generalization. In general, this reflects the advantages of our IMSE to deal with domain shift and task generalization in FS-UDA, especially for large domain gap.

**Evaluation on FSL methods with our domain alignment modules.** To further validate the efficacy of the two alignment modules designed in our IMSE, i.e., similarity pattern alignment and local descriptor alignment, we incorporate their corresponding loss terms \( \text{Lsym}, \text{Ladv}, \text{Lmsn} \) into the loss functions of three FSL models ProtoNet, DeepEMD and DN4, namely IMSE+ProtoNet, IMSE+DeepEMD and IMSE+DN4, respectively. We test them on 3000 new FS-UDA tasks, compared with the FSL methods (ProtoNet, DeepEMD and DN4) and the combination methods with ADDA [29] (ADDA+ProtoNet, ADDA+DeepEMD and ADDA+DN4). For simplification and clarification, we average the accuracy from every domain to the other four domains, which are shown in Figure 3. Obviously, these methods IMSE+ProtoNet, IMSE+DeepEMD and IMSE+DN4 are more significantly improved by using our alignment modules in IMSE, which indicates the effectiveness of aligning the SPs and LDs again. Moreover, these methods are still outperformed by our IMSE, because we have a complete and self-contained FS-UDA framework, and the designed similarity patterns are effective for classification. This shows the efficacy of our IMSE again.

**Different losses.** We conduct various experiments on DomainNet to further evaluate the effect of our different losses in Eq.(5). Besides the classification loss (\( \text{Lcls} \)), we combine the remaining
Figure 3: Comparison of applying different domain alignment modules to three FSL methods ProtoNet, DeepEMD and DN4. The red lines denote the accuracy of our method.

Table 5: Ablation study of the three losses designed in our IMSE, where the FS-UDA tasks are evaluated from a source domain (skt) to four different target domains (rel, qdr, pnt, and cli) in DomainNet.

| Components | Target Domains | 1-shot | 5-shot |
|------------|----------------|--------|--------|
| \(L_{\text{ado}}\) | rel | qdr | pnt | cli |
| ✓ | 54.42 | 42.28 | 46.80 | 56.37 |
| ✓ | 52.11 | 44.51 | 48.76 | 56.99 |
| ✓ | 54.53 | 42.10 | 47.62 | 56.58 |
| ✓ | 56.12 | 48.37 | 49.44 | 58.78 |
| ✓ | 55.30 | 45.52 | 49.33 | 58.34 |
| ✓ | 56.57 | 49.03 | 50.17 | 59.12 |
| ✓ | 56.00 | 47.93 | 50.13 | 58.33 |
| ✓ | 57.21 | 49.71 | 50.36 | 59.30 |

three loss terms: 1) LD adversarial training loss \(L_{\text{ado}}\), 2) multiscale LD matching loss \(L_{\text{msm}}\), and 3) similarity pattern alignment loss \(L_{\text{spa}}\). We evaluate both 1-shot and 5-shot FS-UDA tasks by taking sketch as the source domain, and the other four domains as the target domains, respectively. The accuracies on the four target domains are reported in Table 5. As observed, the more the number of loss terms involved, the higher the accuracy. The combination of all the three losses is the best. For the single loss, \(L_{\text{spa}}\) performs better than \(L_{\text{ado}}\) and \(L_{\text{msm}}\), showing the efficacy of SP alignment. The combination of \(L_{\text{ado}}\) and \(L_{\text{msm}}\) is considerably better than either one of them, showing the positive effect of combining multiscale LD matching and LD adversarial training. Based on the above, the addition of \(L_{\text{spa}}\) further improves the performance, indicating the necessity of jointly aligning SPs and LDs.

Gaussian kernel and maxpooling for SPs. We further investigate the process of generating our SPs by applying Gaussian kernel and maxpooling (see Figure 1). We try to change the filters and the poolings, and compare the variants in Table 4. As seen, our method in the first row performs best. The performance decreases sharply when Gaussian filter is replaced or removed (see rows 3-4 in Table 4). This is because Gaussian kernel efficiently makes the similarity of adjacent points smooth. For downsampling in the second row, we remove the maxpooling and perform the downsampling by changing the filter stride to 2, but it performs worse than our IMSE using maxpooling in the first row. This is because maxpooling extracts local maximum responses of similarity maps that could reflect the prominent and discriminative characteristics for classification.

Cross-dataset evaluation. We investigate the performance of our model on a substantially different dataset miniImageNet, after episodic training on DomainNet. To produce two domains for FS-UDA, we modify miniImageNet by transferring a half of real images (rel) into sketch images (skt) by MUNIT [11]. We conduct experiments between rel and skt, and compare our IMSE with DN4 and DeepEMD for 5-way 1-shot UDA tasks. Their accuracies are shown in Table 6. As seen, our method still performs better than DN4 and DeepEMD, showing the efficacy of our IMSE again. Compared with DomainNet, the results in miniImageNet have some performance degeneration, since their sketch images are more simplified than that in DomainNet, and thus have larger difference from real images.

Hyper-parameters sensitive analysis. To investigate the effect of three domain alignment losses, we conduct the sensitive experiments of three hyper-parameters in Eq. (5), i.e. \(\lambda_a\), \(\lambda_s\) and \(\lambda_m\), respectively. We fix two parameters as the optimal values, and meanwhile change the other one for performance evaluation. The changing accuracy curves of the three parameters are shown in Figure 4. Obviously, the curves of the parameters \(\lambda_s\) and \(\lambda_m\) are stably changing. When \(\lambda_a\) is less than 0.1, it also changes slowly. This shows the low sensitivity of the three parameters.

4.2 Few-shot learning on miniImageNet

Additionally, we investigate the performance of our IMSE for few-shot learning with 1-shot and 5-shot setting on a benchmark dataset miniImageNet. We adopt the same data partitioning, training and testing setting as in [13]. Since in few-shot learning the support and query sets are from the same domain, only one domain exists and no domain alignment is required in this experiment. Thus, we discard the LD-based domain alignment losses \(L_{\text{ado}}\) and \(L_{\text{msm}}\) used in our method, and retain the classification loss \(L_{\text{cls}}\). Meanwhile, we modify the similarity pattern (SP) alignment loss \(L_{\text{spa}}\) to constrain the covariance matrix of SPs to be an identity matrix: \(L_{\text{spa}}(\Sigma) = \frac{1}{NK} \sum_{i,j \in K} \| \Sigma_i - \Sigma_j \| _{F}^2 \). where \(\Sigma = \frac{1}{NK} \text{tr}(\Sigma^2)\). It can be viewed as a regularizer to learn less biased SPs for classification.

We compare our method with seven representative FSL methods in Table 7. The results of almost all the compared methods
are quoted from [35], except CAN and DN4. The results of CAN are quoted from its original work [9]. Since DN4 in [13] did not use backbone ResNet-12 (our setting), we use its released code, the backbone ResNet-12 and pretrained to do experiments for a fair comparison. Note that, we compare the FC version of DeepEMD, because it has no data augmentation, neither ours. Also, we evaluate IMSE by using the loss $L_{rspa}$ or not. As observed in Table 7, IMSE performs better than most of recent FSL methods, showing excellent performance of IMSE for FSL. Meanwhile, our method is slightly lower than DeepEMD, especially for 5-shot. This may be because DeepEMD needs to learn new prototypes for each testing 5-shot task, which could improve the performance but is more time-consuming than ours. In addition, in our IMSE, it is much better to use the $L_{rspa}$ than to remove it, which indicates the positive effect of using covariance matrix of SPs as regularization.

5 CONCLUSION

In this paper, we propose a novel method IMSE for the new setting of FS-UDA. Our IMSE leverages local descriptors to design the similarity patterns to measure the similarity of query images and support classes from different domains. To learn a domain-shared embedding model for FS-UDA, we align the local descriptors by adversarial training and multi-scale matching, and align the covariance matrices of the similarity patterns. Extension experiments over DomainNet show the efficacy of our IMSE for FS-UDA, and our method also has good performance for FSL on miniImageNet.

ACKNOWLEDGMENTS

Wanqi Yang and Ming Yang are supported by National Natural Science Foundation of China (Grant Nos. 62076135, 61876087). Lei Wang is supported by an Australian Research Council Discovery Project (No. DP200101289) funded by the Australian Government.

REFERENCES

[1] David Berthelot, Tom Schumm, and Luke Metz. 2017. BEGAN: Boundary Equilibration Generative Adversarial Networks. CoRR (2017). arXiv:1703.10717
[2] Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. 2019. Meta-learning with Differentiable Closed-form Solvers. In ICLR.
[3] Yue Cao, Ming-Sheng Long, and Jianmin Wang. 2018. Unsupervised Domain Adaptation with Distribution Matching Machines. In AAAI. 2795–2802.
[4] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. 2019. A Closer Look at Few-shot Classification. In ICLR.
[5] Yinbo Chen, Xiaolong Wang, Zhong Liu, Huijuan Xu, and Trevor Darrell. 2020. A New Meta-baseline for Few-shot Learning. CoRR (2020). arXiv:2003.04390
[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic Meta-learning for Fast Adaptation of Deep Networks. In ICML. 1126–1135.
[7] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial Training of Neural Networks. J. Mach. Learn. Res. 17, 1 (2016), 2096–2030.
[8] Ran He, Xiang Wu, Zhenan Sun, and Tieniu Tan. 2019. Wasserstein CNN: Learning Invariant Features for NIR-VIS Face Recognition. Trans. Pattern Anal. Mach. Intell. (2019). 1761–1773.
[9] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. 2019. Cross Attention Network for Few-shot Classification. In NIPS. 4005–4016.
[10] Langbing Hu, Meina Kan, Shiguang Shan, and Xilin Chen. 2020. Unsupervised Domain Adaptation with Hierarchical Gradient Synchronization. In CVPR. 4042–4051.
[11] Xin Huang, Ming-Yu Liu, Serge J. Belongie, and Jan Kautz. 2018. Multimodal Unsupervised Image-to-Image Translation. In ECCV. 172–189.
[12] Wenbin Li, Lei Wang, Jing Huo, Yinghuan Shi, Yang Gao, and Jiebo Luo. 2020. Asymmetric Distribution Measure for Few-shot Learning. In IJCAI. 2957–2963.
[13] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. 2019. Revisiting Local Descriptor based Image-to-class Measure for Few-shot Learning. In CVPR. 7260–7268.
[14] Wenbin Li, Jinglin Xu, Jing Huo, Lei Wang, Yang Gao, and Jiebo Luo. 2019. Distribution Consistency based Covariance Metric Networks for Few-shot Learning. In AAAI. 8642–8649.
[15] Yann LeCun, Yann Avritha, Sylvaine Picard, and Andrei Bursuc. 2019. Dense Classification and Implanting for Few-shot Learning. In CVPR. 9258–9267.
[16] Mingzhang Long and Jianmin Wang. 2015. Learning Transferable Features with Deep Adaptation Networks. In ICLR. 97–105.
[17] Yuwei Luo, Ping Liu, Tao Guan, Junqing Yu, and Yi Yang. 2020. Adversarial Style Mining for One-shot Unsupervised Domain Adaptation. In NIPS.
[18] Saurav Motian, Quinn Jones, Seyed Imrananesh, and Gianfranco Doretto. 2017. Few-shot Adversarial Domain Adaptation. In NIPS. 6670–6680.
[19] Boris N. Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. 2018. TADAM: Task Dependent Adaptive Metric for Improved Few-shot Learning. In NIPS. 719–729.
[20] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. 2019. Moment Matching for Multi-source Domain Adaptation. In IJCAI. 1406–1415.
[21] Sachin Ravi and Hugo Larochelle. 2017. Optimization as a Model for Few-shot Learning. In ICLR.
[22] Subhankar Roy, Aliaksandr Siarohin, Enver Sangineto, Samuel Rota Bulò, Nicu Sebe, and Elisa Ricci. 2019. Unsupervised Domain Adaptation using Feature-whitening and Consensus Loss. In CVPR. 9471–9480.
[23] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Maximum Classifier Discrepancy for Unsupervised Domain Adaptation. In CVPR. 3723–3732.
[24] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In NIPS. 4077–4087.
[25] Quanru Sun, Yaoao Liu, Tat-Seng Chua, and Bernt Schiele. 2019. Meta-transfer Learning for Few-shot Learning. In CVPR. 403–412.
[26] Hui Tang, Ke Chen, and Kui Jia. 2020. Unsupervised Domain Adaptation via Structurally Regularized Deep Clustering. In CVPR. 8722–8732.
[27] Takeshi Teshima, Isei Sato, and Masashi Sugiyama. 2020. Few-shot Domain Adaptation by Causal Mechanism Transfer. In IJCAI. 9258–9267.
[28] Arnold, Preethi, and Geoffrey W. Hinton. 2019. Domain Adaptation with Hierarchical Gradient Synchronization. In CVPR. 4042–4051.
[29] Wenbin Li, Lei Wang, Jing Huo, Yinghuan Shi, Yang Gao, and Jiebo Luo. 2020. Asymmetric Distribution Measure for Few-shot Learning. In IJCAI. 2957–2963.
[30] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep Domain Confusion: Maximizing for Domain Invariance. In ICML. 7260–7268.
[31] Boris N. Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. 2018. TADAM: Task Dependent Adaptive Metric for Improved Few-shot Learning. In NIPS. 719–729.
[32] Qian Wang and Toby P. Breckon. 2020. Unsupervised Domain Adaptation via Causal Mechanism Transfer. In CVPR. 9258–9267.
[33] Xiang Xu, Xiong Zhou, Ragav Venkatesan, Gurumurthy Swaminathan, and Or- chid Majumder. 2019. d-SNE: Domain Adaptation using Stochastic Neighborhood Embedding. In CVPR. 3723–3732.
[34] Xiang Xu, Xiong Zhou, Ragav Venkatesan, Gurumurthy Swaminathan, and Or- chid Majumder. 2019. d-SNE: Domain Adaptation using Stochastic Neighborhood Embedding. In CVPR. 3723–3732.
[35] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. 2020. DeepEMD: Few-shot Learning via Embedding Adaptation with Set-to-set Functions. In CVPR. 9258–9267.