Collaborative Optimization and Aggregation for Decentralized Domain Generalization and Adaptation

Guile Wu and Shaogang Gong
Queen Mary University of London
{guile.wu, s.gong}@qmul.ac.uk

Abstract

Contemporary domain generalization (DG) and multi-source unsupervised domain adaptation (UDA) methods mostly collect data from multiple domains together for joint optimization. However, this centralized training paradigm poses a threat to data privacy and is not applicable when data are non-shared across domains. In this work, we propose a new approach called Collaborative Optimization and Aggregation (COPA), which aims at optimizing a generalized target model for decentralized DG and UDA, where data from different domains are non-shared and private. Our base model consists of a domain-invariant feature extractor and an ensemble of domain-specific classifiers. In an iterative learning process, we optimize a local model for each domain, and then centrally aggregate local feature extractors and assemble domain-specific classifiers to construct a generalized global model, without sharing data from different domains. To improve generalization of feature extractors, we employ hybrid batch-instance normalization and collaboration of frozen classifiers. For better decentralized UDA, we further introduce a prediction agreement mechanism to overcome local disparities towards central model aggregation. Extensive experiments on five DG and UDA benchmark datasets show that COPA is capable of achieving comparable performance against the state-of-the-art DG and UDA methods without the need for centralized data collection in model training.

1. Introduction

Deep neural networks have advanced significantly over the past decade and achieved promising performance for many visual recognition tasks. However, due to the presence of data bias [33] between training and test data (*a.k.a.* domain shift [28]), models elaborately optimized with labeled training data from some source domains usually suffer from significant performance degradation on new target domains. To resolve this problem, domain generalization (DG) [52, 40, 15] and unsupervised domain adaptation (UDA) [35, 38, 26] have been intensively studied, which aim at generalizing models learned on source domains to new target domains.

Traditionally, DG and UDA can use a single source domain for generalization learning by adversarial data augmentation or domain alignment [37, 25]. But real-world data are usually collected from different domains under different conditions (*e.g.*, style and environment), so recent studies focus more on multi-source DG [15] and UDA [26], yielding better generalization performance. In this work, we also focus on multi-source DG and UDA. Contemporary multi-source DG [31, 47, 50] and UDA [2, 26, 51] share the assumption that training data are collected from multiple domains to jointly optimize a generalized model. While DG focuses on direct deployment on unseen new target domains, UDA utilizes unlabeled data from target domains to further reduce domain discrepancy. However, this centralized model learning paradigm is not applicable when source data from different domains cannot be shared for joint training, due to either data privacy or storage/transmission limitations. To address this problem, several recent studies [27, 7] resort to federated learning [21, 12] for developing decentralized UDA by federated adversarial training [27] or knowledge distillation [7]. However, these methods rely heavily on unlabeled target domain data for learning a global model and fail to resolve the more challenging decentralized DG problem.

In this work, we study the problems of decentralized DG and UDA, which aim at optimizing a generalized target model via decentralized learning with non-shared data from multiple domains. To this end, we propose a new approach called Collaborative OPtimization and Aggregation (COPA). Fig. 1 illustrates our approach. For decentralized DG (*steps 1, 3 and 5*): In each source domain (*step 1*), we optimize a local model, which consists of a domain-invariant feature extractor and an ensemble of domain-specific classifiers, using non-shared and private local training data. Next (*step 3*), we centrally aggregate local feature extractors as a global domain-invariant feature

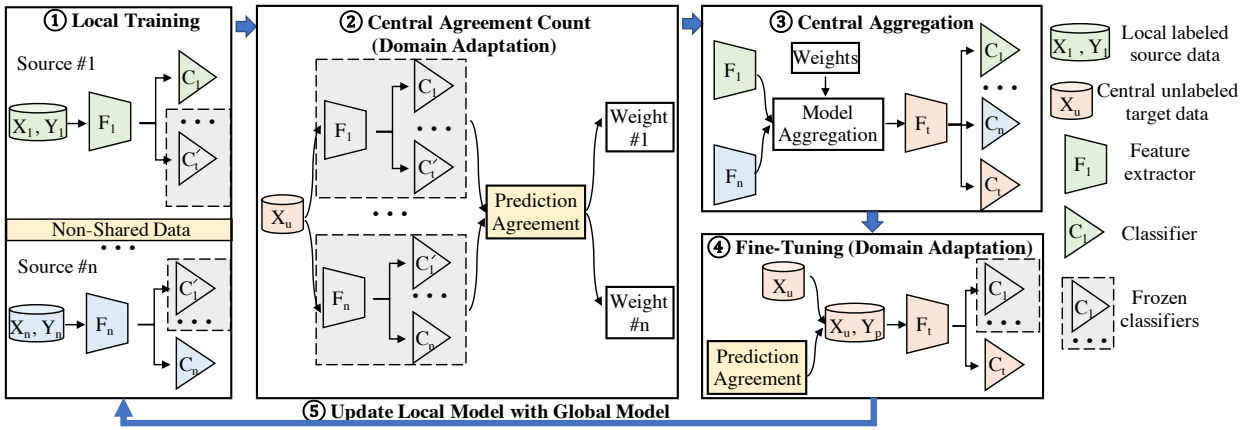

Figure 1. An overview of the proposed Collaborative Optimization and Aggregation (COPA) approach to decentralized DG and UDA. In decentralized DG, steps 1, 3 and 5 are iteratively performed, while in decentralized UDA, steps 1-5 are iteratively performed (with central unlabeled data from a target domain). Note that source labeled data from different domains are non-shared and C_t only exists for UDA.

extractor and assemble domain-specific classifiers as an ensemble of classifiers. Together they represent a generalized global model which is then used to update local models to facilitate the collaborative optimization (*step 5*). For decentralized UDA (*steps 1-5*): Given additional unlabeled training data from a target domain, we measure centrally the prediction agreement among local models (*step 2*) to generate weights for model aggregation and pseudo labels for model fine-tuning (*step 4*). This collaborative optimization and aggregation process is iterative, and it allows us to learn a generalized global model for both decentralized DG and UDA without sharing local data across domains.

Contributions. We propose a new approach called Collaborative Optimization and Aggregation (COPA) to resolve both decentralized DG and UDA problems. This differs from the conventional centralized DG and UDA methods [31, 51, 26] that collect data from different domains together for joint training. We optimize domain-invariant feature extractors for central aggregation and domain-specific classifiers for central ensembling. This approach enables more selective knowledge disentanglement between domain-invariant feature representations and domain-specific classification information. This differs from aggregating all parameters indiscriminately of local models for constructing a global model [21, 27, 7]. For better decentralized UDA, we further introduce a prediction agreement mechanism to facilitate central model aggregation. We conduct extensive experiments on five DG and UDA benchmark datasets and show that COPA decentralized learning approach is capable of achieving comparable performance against the state-of-the-art methods.

2. Related Work

Domain Generalization and Unsupervised Domain Adaptation. Domain generalization (DG) aims to generalize a model learned on source domains to any unseen target domains. Prevailing approaches include learning domain-invariant representations by aligning source domain data distributions [16], optimizing domain-specific normalization [31], synthesizing data to augment source domains [50], employing episodic training to improve robustness of a network against domain shift [15], *etc*. Unsupervised domain adaptation (UDA) is closely related to DG but with unlabeled target domain data to bridge the source and target domains. One of the most popular approaches for UDA is to reduce distribution divergence based on Maximum Mean Discrepancy (MMD) [3, 18]. Another promising approach is to use adversarial training [45, 34] for feature alignment across source and target domains. Since DG and UDA are closely related, some studies [22, 51] also propose to develop a unified framework for both DG and UDA. For example, Zhou *et al*. [51] employ domain adaptive ensemble learning with consistency regularization to build a unified framework (DAEL) for both DG and UDA.

Our approach (COPA) differs significantly from the conventional centralized DG and UDA methods in that: (1) COPA protects data privacy by decentralized learning without sharing source data across domains, while most existing methods use a centralized training paradigm without privacy concern; (2) COPA optimizes a local model in each domain using non-shared training data and centrally aggregates local models to construct a generalized global model, instead of using a shared model for joint optimization with collected data from different domains [31, 52, 44, 35]; (3) COPA improves generalization of a domain-invariant feature extractor with hybrid batch-instance normalization layers [20] and collaboration of multiple classifiers, instead of episodic training [15] or consistency regularization [51].

Federated Learning. Federated learning [21, 12, 39] is a distributed learning paradigm for optimizing a central

model with the collaboration of multiple local client models without sharing local data, enabling decentralized learning in a privacy-preserving way. FedAvg [21] is one of the most popular methods to implement federated learning, which iteratively averages local model updates to construct a central model. This method is simple but effective and can improve communication efficiency. The proposed COPA shares the principle of FedAvg, but: (1) Instead of aggregating all parameters of local models to construct a global model, we optimize domain-invariant feature extractors for more selective central aggregation and domain-specific classifiers for central ensembling; (2) Source data in COPA are from different domains with domain shift, instead of different data partitions from the same dataset.

Decentralized Learning from Multiple Domains. With the awareness of data privacy, some recent works [27, 41, 7] have studied decentralized learning from multiple domains in visual recognition. In [27], a federated adversarial domain adaptation method with feature disentanglement is proposed to resolve domain shift for decentralized UDA. In [41], a federated person re-identification method is introduced to optimize a generalizable embedding model by knowledge distillation and model aggregation. In [7], a knowledge distillation based federated learning method is presented to decentralized UDA. Our approach differs from these methods in that: (1) Instead of aggregating all parameters of local models [27, 7] or learning a generic central feature embedding model [41], we optimize domain-invariant feature extractors for central aggregation and domain-specific classifiers for central ensembling, for better disentanglement and selection of knowledge transfer among different domains; (2) We improve generalization of feature extractors with hybrid batch-instance normalization layers and collaboration of domain-specific classifiers, instead of knowledge distillation [41, 7]; (3) We introduce a prediction agreement mechanism to facilitate collaborative model optimization for decentralized UDA, instead of only averaging models to learn a generic global embedding model [41].

3. Methodology

Problem Statement. In this work, we aim to optimize a generalized model with non-shared data from multiple domains for decentralized DG and UDA. Given n source domains $(\{X_1, Y_1\}, ..., \{X_n, Y_n\})$, where each domain $\{X_i, Y_i\}$ contains N_i labeled samples X_i for K classes and Y_i=$\{1, ..., K\}$, we maintain *separately* source data from each domain (local and non-shared) for local model training. In decentralized DG, a generalized global model is optimized without sharing local source data nor using any unlabeled data from target domains for fine-tuning. In decentralized UDA, a global model also cannot access local source data but has unlabeled samples X_u from a target domain for central fine-tuning.

3.1. Approach Overview

We depict an overview of the proposed Collaborative Optimization and Aggregation (COPA) in Fig. 1, where decentralized DG is accomplished with steps 1, 3 and 5, while decentralized UDA is accomplished with steps 1-5. Specifically, with $\{X_i, Y_i\}_{i=1}^n$, we train local models $\{F_i, C_i\}$ for each domain (*step 1*), where F_i and C_i are the feature extractor and the classifier for the i-th domain. After training local models for m local epochs, we aggregate parameters of $\{F_i\}_{i=1}^n$ centrally to construct a global domain-invariant feature extractor F_t and assemble $\{C_i\}_{i=1}^n$ centrally to build an ensemble of domain-specific classifiers (*step 3*), where $\{F_t, \{C_i\}_{i=1}^n\}$ forms a global model. The global model is then used to update local models to facilitate local optimization (*step 5*). This process is iteratively performed for g global iterations (*iterate steps 1, 3 and 5*) to optimize a generalized global model for deployment. In decentralized UDA, with unlabeled data X_u from a target domain, we use additionally X_u to measure the prediction agreement among local models (*step 2*) to perform weighted aggregation of $\{F_i\}_{i=1}^n$ and to generate pseudo labels $\{Y_u\}$ for fine-tuning $\{F_t, C_t\}$ (*step 4*). Thus, the iterative learning process for UDA consists of steps 1-5, resulting in a global model $\{F_t, \{C_i\}_{i=1}^n, C_t\}$.

3.2. Local Model Collaborative Optimization

In each source domain, with labeled training data $\{X_i, Y_i\}$, we use a cross-entropy loss $\mathcal{L}_{ce}(X_i, Y_i)$ to optimize $\{F_i, C_i\}$, formulated as:

$$\mathcal{L}_{ce}(X_i, Y_i) = \frac{1}{N_i} \sum_{x \in X_i, y \in Y_i} \ell_{ce}(y, C_i(F_i(x))), \quad (1)$$

where $\ell_{ce}(\cdot, \cdot)$ is a cross-entropy loss function. However, different from the conventional centralized training paradigm, $\{X_i, Y_i\}_{i=1}^n$ are non-shared across domains in decentralized learning, so we cannot jointly optimize a model with data from different domains. As a result, each local model only learns domain-specific information. To alleviate this problem, we filter out domain-specific information in F_i with hybrid batch-instance normalization layers and further improve the generalization of F_i via collaboration of frozen classifiers $\{C'_j\}_{j=1, j \neq i}^n$.

Learning Domain-Invariant Representation. Instance normalization has shown the effectiveness for filtering out domain-specific information [24, 20, 31]. However, directly using instance normalization in lieu of batch normalization will lose useful statistic information learned by batch normalization, resulting in significant performance degradation in visual recognition [24]. In the light of this, we use hybrid batch-instance normalization layers [20, 31] to replace

batch normalization layers in feature extractors. Specifically, we combine batch normalization with instance normalization as:

$$\hat{h} = \gamma \frac{h - (\omega_{bn}\mu_{bn} + \omega_{in}\mu_{in})}{\sqrt{\omega'_{bn}\sigma^2_{bn} + \omega'_{in}\sigma^2_{in} + \epsilon}} + \beta, \tag{2}$$

where $h, \hat{h} \in \mathbb{R}^{B\times E\times W\times H}$ are activations (a 4D tensor with batch size B, channel number E, width W and height H), γ and β are affine parameters, ϵ=1e−5 is for numerical stability, μ and σ^2 are means and variances (defined as Eq. (3)), ω and ω' are ratios to weight the mixture of means and variances for batch normalization and instance normalization (defined as Eq. (4)).

$$\begin{aligned} \mu_{in} &= \frac{\sum_{i,j}^{H,W} h_{b,i,j}}{HW},\ \sigma^2_{in} = \frac{\sum_{i,j}^{H,W}(h_{b,i,j} - \mu_{in})^2}{HW}, \\ \mu_{bn} &= \frac{\sum_{b=1}^{B} \mu_{in}}{B},\ \sigma^2_{bn} = \frac{\sum_{b=1}^{B}(\sigma^2_{in} + \mu^2_{in}) - \mu^2_{bn}}{B}. \end{aligned} \tag{3}$$

$$\begin{aligned} \omega_i &= \frac{e^{\lambda_i}}{\sum_{j\in\{in,bn\}} e^{\lambda_j}},\ \text{and}\ \ i\in\{in, bn\}, \\ \omega'_i &= \frac{e^{\lambda'_i}}{\sum_{j\in\{in,bn\}} e^{\lambda'_j}},\ \text{and}\ \ i\in\{in, bn\}, \end{aligned} \tag{4}$$

where λ and λ' are learnable parameters. Note that different from [31] which learns domain-specific hybrid normalization layers for DG, we resolve domain shift (Eq. (2)) by learning domain-invariant feature extractors for central aggregation while encoding domain-specific information into the assembled classifiers.

Collaboration of Frozen Classifiers. As each domain is non-shared, we cannot use data from different domains to jointly optimize a shared feature extractor. However, with an ensemble of domain-specific classifiers, we can improve generalization of a feature extractor F_i via encouraging F_i to generate domain-invariant representations for 'new' classifiers and 'new' data. Specifically, in our design, both the global model and local models are using a multi-head architecture. Thus, as shown in Fig. 1, when receiving updates from a global model, a local model simultaneously gets other domain-specific classifiers, *i.e.*, a local model is updated as $\{F_i, C_i, \{C'_j\}^n_{j=1,j\neq i}\}$, where $\{C'_j\}^n_{j=1,j\neq i}$ are frozen classifiers from other domains. Here, $\{C'_j\}^n_{j=1,j\neq i}$ are frozen because we do not have data from other domains to train these domain-specific classifiers, but since they are 'new' to F_i, they can be used to encourage F_i to generate domain-invariant features. This shares the merit of episodic training for DG [15] but in a decentralized training paradigm instead of episodic training.

However, since F_i is simultaneously optimized with a domain-specific C_i, directly putting the same feature representations through C_i and $\{C'_j\}^n_{j=1,j\neq i}$ will result in sub-optimal performance. To address this problem, we use RandAugment [4] to augment input samples $A(X_i)$ and compute a cross-entropy loss $\mathcal{L}'_{ce}$ for each frozen classifier (similar to Eq. (1)), while C_i is still trained with X_i under standard augmentation. Note that, different from directly enlarging datasets with data augmentation and [51] that uses RandAugment for optimization with consistency regularization, we use RandAugment to generate 'new' samples for F_i and use the frozen $\{C'_j\}^n_{j=1,j\neq i}$ to improve generalization of domain-invariant F_i with $\mathcal{L}'_{ce}$ (see experiments § 4.3 for evaluation). Thus, the training objective of the i-th local model is formulated as:

$$\mathcal{L}_i = \mathcal{L}_{ce}(C_i; X_i, Y_i) + \sum_{j=1,j\neq i}^{n} \mathcal{L}'_{ce}(C'_j; A(X_i), Y_i). \tag{5}$$

3.3. Global Model Optimization and Aggregation

After training local models for m local epochs, we perform central aggregation to learn a global model. In decentralized DG, there are neither unlabeled data from target domains nor centrally collected data from source domains, so we do not fine-tune a global model. In decentralized UDA, there are unlabeled data from a target domain for learning prediction agreement and fine-tuning the model.

Decentralized DG without Unlabeled Target Data. A simple way for central aggregation is averaging all parameters of local models as FedAvg [21]. However, this uniform aggregation approach impairs the optimization of domain-invariant feature extractors and domain-specific classifiers. To resolve this problem, as shown in Fig. 1 (*step 3*), we aggregate model parameters $\{\Theta_i\}^n_{i=1}$ of $\{F_i\}^n_{i=1}$ to construct a global domain-invariant feature extractor F_t with model parameter Θ_t (formulated as Eq. (6)) and assemble domain-specific classifiers to create an ensemble $\{C_i\}^n_{i=1}$. Thus, a global model is formulated as $\{F_t, \{C_i\}^n_{i=1}\}$.

$$\Theta_t \leftarrow \sum_{i=1}^{n} \alpha_i \cdot \Theta_i, \tag{6}$$

where $\alpha_i \in [0, 1]$ are weights for aggregating F_i and $\sum_i^n \alpha_i = 1$. Setting $\alpha_i = \frac{1}{n}$ indicates that local models are equally important for central aggregation. Here, a white noise [9] can be added to Eq. (6) to further protect privacy against attack. We then use a global model to re-initialize each local model, which helps to indirectly incorporate knowledge from other domains into each local model and to facilitate subsequent model aggregation. We iteratively optimize local models and a global model (Fig. 1 steps 1, 3 and 5).

Decentralized UDA with Unlabeled Target Data. As shown in Fig. 1, in decentralized UDA, we add steps 2 and 4 to the iterative learning process to leverage unlabeled target data for better aggregation and optimization. Specifically,

when local models are collected, we use them to respectively generate prediction p_i for each sample x in X_u as:

$$p_i(x) = \frac{1}{n}\big(C_i(F_i(x)) + \sum_{j=1,j\neq i}^{n} C'_j(F_i(x))\big). \quad (7)$$

This prediction is the sum of predictions of an ensemble, so it is more reliable and can be used to generate pseudo labels y for x. There are three ways to generate a pseudo label of x based on $\{p_i(x)\}_i^n$: (1) use the most confident prediction (maximum probability prediction) among $\{p_i(x)\}_i^n$; (2) compute the mean of $\{p_i(x)\}_i^n$ and use the maximum probability prediction; (3) compute the maximum probability prediction of each $\{p_i(x)\}_i^n$ and use the most frequent prediction. In practice, we only generate pseudo labels if the maximum probability is larger than a threshold (*e.g.*, 0.95). Meanwhile, we count the number of prediction agreement (Z_i) for each local model: If a prediction of the i-th model is consistent with a pseudo label y, then we add 1 to Z_i, which indicates a correct prediction of the i-th model. After measuring prediction agreement with all samples in X_u, we compute the aggregation weight α_i as:

$$\alpha_i = \frac{e^{\bar{Z}_i}}{\sum_{j=1}^{n} e^{\bar{Z}_j}}, \text{ and } \bar{Z}_i = \frac{Z_i}{\sum_{j=1}^{n} Z_j}. \quad (8)$$

With Eq. (8), α_i reflects the generalization of F_i for a target domain, which facilitates better aggregation in Eq. (6).

After central aggregation, we further use X_u with pseudo labels Y_u to fine-tune $\{F_t, C_t\}$ with a cross-entropy loss, where C_t is a domain-specific classifier for the target domain. Similar to local model training, the frozen classifiers $\{C'_i\}_{i=1}^n$ are also used to improve the generalization of F_t with augmented samples and cross-entropy losses. Thus, the training objective $\mathcal{L}_t$ for central fine-tuning is similar to Eq. (5). Note that, compared with decentralized DG, both local and global models in decentralized UDA have an additional classifier C_t to further facilitate model learning.

Summary. With iteratively collaborative optimization and aggregation between local models and the global model, COPA accomplishes decentralized DG and UDA without collecting source data together for joint training. We summarize the training process of COPA for decentralized DG in Algorithm 1 and for decentralized UDA in Algorithm 2.

Algorithm 1 The proposed COPA for decentralized DG.

Input: n source domains $\{X_i, Y_i\}_{i=1}^n$, n local models $\{F_i, C_i, \{C'_j\}_{j=1,j\neq i}^n\}_{i=1}^n$, a global model $\{F_t, \{C_i\}_{i=1}^n\}$.

for G=1:g **do** /*Global iteration*/
  for i=1:n **do** /*i-th local source domain*/
    Update local model with global model (*step 5*)
    for M=1:m **do** /*Local training (*step 1*)*/
      Compute $\mathcal{L}_{ce}$ by $C_i(F_i(X_i))$
      Compute $\mathcal{L}'_{ce}$ by $\sum_{j=1,j\neq i}^{n} C'_j(F_i(A(X_i)))$
      Train local model with $\mathcal{L}_i$ (Eq. (5))
  end for
  Construct F_t by aggregation (Eq. (6)) (*step 3*)
  Assemble $\{C_i\}_{i=1}^n$ for classification
end for

Output: A generalized global model $\{F_t, \{C_i\}_{i=1}^n\}$.

Algorithm 2 The proposed COPA for decentralized UDA.

Input: n source domains $\{X_i, Y_i\}_{i=1}^n$, n local models $\{F_i, C_i, \{C'_j\}_{j=1,j\neq i}^n, C'_t\}_{i=1}^n$, a global model $\{F_t, \{C_i\}_{i=1}^n, C_t\}$, an unlabeled target domain X_u.

for G=1:g **do** /*Global iteration*/
  for i=1:n **do** /*i-th local source domain*/
    Update local model with global model (*step 5*)
    for M=1:m **do** /*Local training (*step 1*)*/
      Compute $\mathcal{L}_{ce}$ by $C_i(F_i(X_i))$
      Compute $\mathcal{L}'_{ce}$ by $\sum_{j=1,j\neq i}^{\{n,t\}} C'_j(F_i(A(X_i)))$
      Train local model with $\mathcal{L}_i$ (Eq. (5))
  end for
  Compute aggregation weight α_i (Eq. (8)) (*step 2*)
  Construct F_t by aggregation (Eq. (6)) (*step 3*)
  Assemble $\{C_i\}_{i=1}^n$ for classification
  Get pseudo labels of X_u
  Fine-tune global model with $\mathcal{L}_t$ (*step 4*)
end for

Output: A generalized global model $\{F_t, \{C_i\}_{i=1}^n, C_t\}$.

4. Experiments

To evaluate the proposed COPA, we conduct extensive experiments on five DG and UDA benchmark datasets.

4.1. Comparison with SOTAs on DG Benchmarks

Datasets. *PACS* [14] is a challenging DG dataset consists of seven object categories from four domains (Art Painting, Cartoon, Photo and Sketch) with large domain discrepancy. *Office-Home* [36] contains about 15,500 images in 65 categories of daily objects from four domains (Artistic, Clipart, Product and Real-World). *Digits-DG* [50] consists of four digit datasets (MNIST [13], MNIST-M [8], SVHN [23] and SYN [8]) with different font style and background. Following previous DG works [50, 31, 51, 40, 15], we adopt the leave-one-domain-out protocol for evaluation by selecting one domain as the unseen new domain for testing while using the remaining domains as source domains for training. But different from the traditional centralized training paradigm, in decentralized DG, each source domain is only used for training a local model, neither mixing with other domains for joint training nor sharing with a central model for fine-tuning. This decentralized approach protects source

Paradigm	Method	art	cat	pho	skt	Avg
Centralized w/o privacy concern	Backbone [51]	77.0	75.9	96.0	69.2	79.5
	Epi-FCR [15]	82.1	77.0	93.9	73.0	81.5
	JiGen [1]	79.4	75.3	96.0	71.4	80.5
	MASF [6]	80.3	77.2	95.0	71.7	81.0
	DGER [47]	80.7	76.4	**96.7**	71.8	81.4
	DAEL [51]	**84.6**	74.4	95.6	78.9	83.4
	L2A-OT [50]	83.3	78.2	96.2	73.6	82.8
	DDAIG [49]	84.2	78.1	95.3	74.7	83.1
	EISNet [40]	81.9	76.4	95.9	74.3	82.2
	DSON [31]	84.7	77.7	95.9	82.2	**85.1**
	MixStyle [52]	84.1	78.8	96.1	75.9	83.7
Decentralized	COPA (ours)	83.3	**79.8**	94.6	**82.5**	**85.1**

Table 1. Comparison with state-of-the-art DG methods on PACS using ResNet-18. Note that centralized methods and decentralized methods are not direct competitors as they use different training paradigms. We report leave-one-domain-out results on Art Painting (art), Cartoon (cat), Photo (pho) and Sketch (skt), and average results of them.

data privacy, although it may result in accuracy degradation.

Implementation Details. On PACS and Office-Home, following [50, 51, 31], we use ResNet-18 [11] pre-trained on ImageNet as the backbone for the feature extractor and a fully connected layer as the classifier. On PACS, we replace all BN layers with hybrid batch-instance normalization layers in the feature extractor, while on Office-Home that has less style variation across domains, we replace the first BN layer and BN layers in the first BasicBlock. Parameters related to batch normalization are initialized with ImageNet pre-trained weights. We use SGD as the optimizer with momentum 0.9 and weight decay $5e$-4. The initial learning rate is set to 0.002, decayed by a cosine annealing rule [19, 51] every global iteration. We set batch size to 30, local epoch m=1, global iteration g=40 and $A(\cdot)$ as RandAugment [4] (with Cutout [5]). On Digits-DG, following [50, 49], we use four 3×3 convolution layers, each of which is followed by ReLU and 2×2 max pooling, as the feature extractor and a fully connected layer as the classifier. We insert a hybrid batch-instance normalization layer after each convolution layer. We use SGD with momentum as the optimizer and set the initial learning rate to 0.05, decayed by 0.1 every 20 global iterations. Training batch size is 30, local epoch m=1, global iteration g=50, and $A(\cdot)$ is RandAugment [4]. We report top-1 accuracy averaged over five runs.

Comparison with the State-of-the-Arts. As shown in Tables 1, 2 and 3, although the proposed COPA uses a decentralized training paradigm for privacy protection, it still achieves comparable performance against the state-of-the-art DG methods that use a centralized training paradigm without privacy concern. Specifically, on PACS (Table 1), COPA outperforms the centralized Backbone that collects all source data for joint training and yields 85.1% average overall accuracy which is on par with the state-of-the-art result. Similarly, on Office-Home (Table 2), COPA achieves 66.1% average overall accuracy, which is superior to the backbone and is still comparable against the state-of-the-art DG methods. On Digits-DG (Table 3), COPA significantly outperforms both the backbone and the state-of-the-arts, achieving the best 81.5% average overall accuracy.

Paradigm	Method	art	clp	prd	rel	Avg
Centralized w/o privacy concern	Backbone [51]	58.9	49.4	74.3	76.2	64.7
	CCSA [22]	59.9	49.9	74.1	75.7	64.9
	CrossGrad [32]	58.4	49.4	73.9	75.8	64.4
	JiGen [1]	53.0	47.5	71.5	72.8	61.2
	DAEL [51]	59.4	**55.1**	74.0	75.7	**66.1**
	L2A-OT [50]	**60.6**	50.1	**74.8**	**77.0**	65.6
	DDAIG [49]	59.2	52.3	74.6	76.0	65.5
	DSON [31]	59.4	45.7	71.8	74.7	62.9
	MixStyle [52]	58.7	53.4	74.2	75.9	65.5
Decentralized	COPA (ours)	59.4	**55.1**	**74.8**	75.0	**66.1**

Table 2. Comparison with state-of-the-art DG methods on Office-Home using ResNet-18. We report leave-one-domain-out results on Artistic (art), Clipart (clp), Product (prd) and Real-World (rel), and average results of them.

Paradigm	Method	mt	mm	sv	sy	Avg
Centralized w/o privacy concern	Backbone [52]	95.8	58.8	61.7	78.6	73.7
	CCSA [22]	95.2	58.2	65.5	79.1	74.5
	CrossGrad [32]	96.7	61.1	65.3	80.2	75.8
	JiGen [1]	96.5	61.4	63.7	74.0	73.9
	L2A-OT [50]	96.7	63.9	68.6	83.2	78.1
	DDAIG [49]	96.6	64.1	68.6	81.0	77.6
	MixStyle [52]	96.5	63.5	64.7	81.2	76.5
Decentralized	COPA (ours)	**97.0**	**66.5**	**71.6**	**90.7**	**81.5**

Table 3. Comparison with state-of-the-art DG methods on Digits-DG using a convolutional backbone [50]. We report leave-one-domain-out results on MNIST (mt), MNIST-M (mm), SVHN (sv) and SYN (sy), and average results of them.

4.2. Comparison with SOTAs on UDA Benchmarks

Datasets. *Digit-Five* [26] is a digit recognition benchmark for UDA, which contains five digit datasets, namely MNIST [13], MNIST-M [8], SVHN [23], SYN [8] and USPS. *Office-Caltech10* [10] consists of 2,533 images of ten object categories from four domains, namely Amazon, Caltech, DSLR and Webcam. Following previous UDA methods [26, 51, 27], we select one domain as the target domain with unlabeled training data, while the other domains are labeled source domains. In decentralized UDA, labeled source domains are non-shared and private while the unlabeled target domain is used for central optimization.

Implementation Details. On Digit-Five, following [26, 48, 51, 7], we use three convolutional layers and two fully connected layers (each of which is followed by a BN layer) as the feature extractor, and use a fully connected layer as the classifier. We replace all 2-D BN layers with hybrid batch-instance normalization layers. We set batch size to

Paradigm	Method	mt	mm	sv	sy	up	Avg
-	Oracle [51]	99.5	95.4	92.3	98.7	99.2	97.0
	SourceOnly[51]	99.1	68.1	84.6	89.9	97.2	87.8
Centralized w/o privacy concern	MDAN [46]	98.0	69.5	69.2	87.4	92.5	83.3
	DCTN [43]	96.2	70.5	77.6	86.8	92.8	84.8
	MCD [30]	99.2	80.7	81.9	95.4	98.3	91.1
	M^{3}SDA [26]	98.4	72.8	81.3	89.6	96.1	87.7
	MME [29]	99.4	83.1	86.4	95.8	98.6	92.7
	DSBN [2]	97.2	71.6	77.9	88.7	96.1	86.3
	MDDA [48]	98.8	78.6	79.3	89.7	93.9	88.1
	LtCMSDA [38]	99.0	85.6	83.2	93.0	98.3	91.8
	CMSS [44]	99.0	75.3	88.4	93.7	97.7	90.8
	DAEL [51]	**99.5**	**93.8**	**92.5**	**97.9**	**98.7**	**96.5**
Decentralized	FADA [27]	91.4	62.5	50.5	71.8	91.7	73.6
	SHOT [17]	98.2	80.2	84.5	91.1	97.1	90.2
	KD3A [7]	99.2	87.3	85.6	89.4	98.5	92.0
	COPA (ours)	**99.4**	**89.8**	**91.0**	**97.5**	**99.2**	**95.4**

Table 4. Comparison with state-of-the-art UDA methods on Digit-Five using a convolutional backbone [26]. We report results on MNIST (mt), MNIST-M (mm), SVHN (sv), SYN (sy) and USPS (up), and average results of them.

Paradigm	Method	A	C	D	W	Avg
-	Oracle [7]	99.7	98.4	99.8	99.7	99.4
	SourceOnly [26]	86.1	87.8	98.3	99.0	92.8
Centralized w/o privacy concern	MDAN [46]	95.4	91.8	98.6	98.9	96.1
	DCTN [43]	92.7	90.2	99.0	99.4	95.3
	MCD [30]	92.1	91.5	99.1	99.5	95.6
	M^{3}SDA [26]	94.5	92.2	99.2	99.5	96.4
	DSBN [2]	93.2	91.6	98.9	99.3	95.8
	CMSS [44]	**96.0**	93.7	99.3	99.6	97.2
	SImpAI [35]	95.6	**94.6**	**100**	**100**	**97.5**
Decentralized	FADA [27]	84.2	88.7	87.1	88.1	87.1
	SHOT [17]	96.4	96.2	98.5	99.7	97.7
	KD3A [7]	**97.4**	**96.4**	98.4	99.7	**97.9**
	COPA (ours)	95.8	94.6	**99.6**	**99.8**	97.5

Table 5. Comparison with state-of-the-art UDA methods on Office-Caltech10 using ResNet-101. We report results on Amazon (A), Caltech (C), DSLR (D) and Webcam (W), and average results of them.

256, local epoch m=1 and global iteration g=30, and use SGD with momentum as the optimizer. The initial learning rate is set to 0.05 with a cosine annealing rule. On Office-Caltech10, following [27, 35, 7], we use ResNet-101 [11] as the feature extractor and a fully connected layer as the classifier. We use hybrid batch-instance normalization layers to replace the first BN layer and BN layers in the first macro block (initialized with ImageNet pre-trained weights). We use batch size 30, local epoch m=1 and global iteration g=40, and use SGD with momentum as the optimizer. The initial learning rate is set to 0.002 with a cosine annealing rule. We report top-1 accuracy averaged over five runs.

Comparison with the State-of-the-Arts. As shown in Tables 4 and 5, the performance of COPA is on par with the state-of-the-art centralized and decentralized UDA methods. Specifically, on Digit-Five (Table 4), COPA significantly outperforms the state-of-the-art decentralized UDA methods and achieves 95.4% average overall accuracy which is close to the best centralized UDA result (96.5%). Since Office-Caltech10 is a small-sized benchmark with only 2,533 images, most state-of-the-art methods achieve close results (Table 5). COPA yields 97.5% average overall accuracy which is comparable against the best centralized result (97.5%) and decentralized result (97.9%). Although COPA is slightly inferior to SHOT [17] and KD3A [7], all three methods yield over 97% average overall accuracy and COPA achieves better results on DSLR and Webcam.

Component	art	cat	pho	skt	Avg
Backbone(sequential)	43.4	65.0	54.6	60.8	55.9
Backbone(centralized) [51]	77.0	75.9	**96.0**	69.2	79.5
Backbone(decentralized w/ FedAvg[21])	76.2	73.8	92.6	68.9	77.9
Backbone(decentralized w/o Iter)	68.9	61.4	92.2	51.7	68.6
COPA	**83.3**	**79.8**	94.6	**82.5**	**85.1**
COPA w/o HBIN	81.4	75.3	95.1	78.5	82.6
COPA w/o CoFC	80.3	77.9	94.9	77.8	82.7
COPA w/o Iter	76.0	72.5	94.3	75.4	79.6
COPA w/o Iter&HBIN&CoFC	69.9	64.6	92.4	51.9	69.7

Table 6. Component effectiveness evaluation on PACS. 'HBIN': Hybrid batch-instance normalization layers. 'CoFC': Collaboration of frozen classifiers. 'w/o Iter': Without iterative optimization and aggregation, *i.e.*, training local models for $m\times g$ epochs and centrally aggregating them once.

4.3. Further Analysis and Discussion

Component Effectiveness Analysis. As shown in Table 6, COPA outperforms the centralized joint-training backbone and the decentralized backbone that iteratively averages all parameters of local models to construct a global model. In local model optimization, without using hybrid batch-instance normalization layers (COPA w/o HBIN) or the collaboration of frozen classifiers (COPA w/o CoFC), the average overall accuracies of COPA decrease by approximately 2% but are still better than both centralized and decentralized backbone models. Besides, we also test the iterative optimization and aggregation mechanism. In Table 6, without iterative optimization, the performance of COPA decreases significantly but is still better than the backbone (decentralized). This further examines the effectiveness of the components for learning domain-invariant representation.

Variants of CoFC component. In addition to the proposed CoFC (Collaboration of Frozen Classifiers), we also evaluate some variants, including: (1) without using CoFC, (2) with CoFC but without using RandAugment, (3) without using CoFC but using RandAugment to enlarge dataset and (4) without using CoFC but using consistency regularization with RandAugment [51]. From Fig. 2, we can see that COPA w/o CoFC yields the worst result and its performance can be improved using RandAug, CR and CoFC. Overall, the proposed CoFC approach yields the best performance.

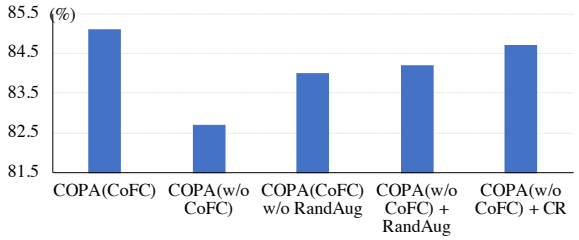

Figure 2. Evaluating variants of the collaboration of frozen classifier mechanism on PACS (average overall accuracy). 'CoFC': Collaboration of frozen classifiers. 'COPA (w/o CoFC) + RandAugment': Directly enlarge dataset using RandAugment without CoFC. 'COPA (w/o CoFC) + CR': Use consistency regularization with RandAugment as [51] for classifiers from other domains and aggregate all learned classifiers for central aggregation.

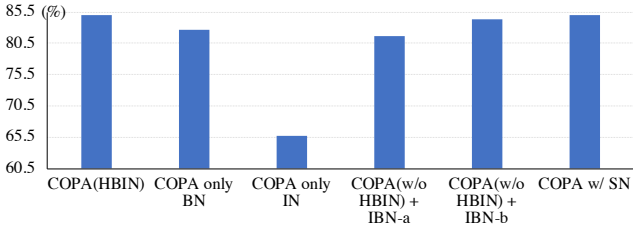

Figure 3. Evaluating variants of normalization in COPA on PACS (average overall accuracy). 'HBIN': Hybrid batch-instance normalization. 'COPA only BN': Without HBIN. 'COPA only IN': Use IN layers to replace all BN layers. 'COPA(w/o HBIN) + IBN-a [24]' and 'COPA(w/o HBIN) + IBN-b [24]': Use IBN to replace HBIN. 'COPA w/ SN': Use SN [20] (HBIN + LayerNorm).

Variants of Normalization. As shown in Fig. 3, we evaluate some variants of learning domain-invariant representation with normalization layers. From Fig. 3, we can see that: (1) COPA w/ HBIN performs closely with COPA w/ SN [20], but SN requires more layer normalization operations; (2) Using only IN in COPA leads to significant performance degradation; (3) COPA w/ IBN [24] can also yield competitive performance but is still inferior to COPA w/ HBIN.

Central Prediction Agreement Analysis. In Fig. 4, we evaluate different central prediction agreement approaches on Digit-Five for decentralized UDA. We can see that COPA(mean) performs better than other approaches on different datasets, except on MNIST where all compared approaches perform closely. Overall, COPA with prediction agreement weights is superior to COPA without using prediction agreement weights, but on SVHN and SYN, COPA(most) performs worse than COPA w/o agree weights. We conjecture that on SVHN and SYN, most local models provide incorrect predictions with low confidence leading to performance degradation in COPA(most), which is resolved when using mean or maximum predictions. By default, we use COPA(mean) for decentralized UDA.

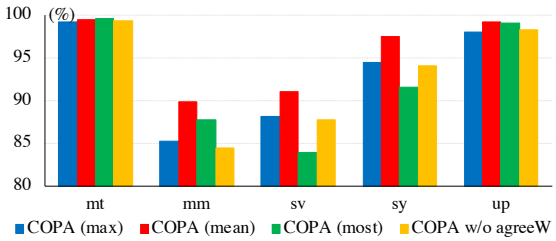

Figure 4. Evaluation of different central prediction agreement approaches on Digit-Five. 'COPA w/o agreeW': Without using prediction agreement weights.

Component	art	cat	pho	skt	Avg
COPA	83.3	**79.8**	94.6	**82.5**	**85.1**
Ind-Ensemble	72.3	55.2	94.7	60.2	70.6
Ind-ParamAvg	68.9	61.4	92.2	51.7	68.6
DAEL [51] (centralized)	**84.6**	74.4	**95.6**	78.9	83.4

Table 7. Comparison with ensemble methods on PACS. 'Ind-Ensemble': Independently train backbone models for each source domain and use the logit ensemble as the prediction. 'Ind-ParamAvg': Independently train backbone models for each source domain and average model parameters as a global model.

Comparison with Ensemble. From Table 7, we can see that COPA performs significantly better than two backbone ensemble methods (Ind-Ensemble and Ind-ParamAvg). When compared with the state-of-the-art centralized ensemble method (DAEL [51]), COPA still achieves comparable performance and yields better average overall accuracy 85.1% on PACS, even though COPA uses a decentralized training paradigm to protect data privacy.

5. Conclusion

In this work, we introduce a new approach called Collaborative OPtimization and Aggregation (COPA) for decentralized domain generalization (DG) and multi-source unsupervised domain adaptation (UDA). The main idea is to iteratively optimize local domain-invariant feature extractors and domain-specific classifiers using non-shared source data for constructing a generalized global model. This allows to accomplish decentralized DG and UDA without sharing data across domains for privacy concern. Extensive experiments on five DG and UDA benchmark datasets demonstrate that COPA is competitive against the state-of-the-art DG and UDA methods. One research direction for future work is to reduce communication cost in the iterative learning process using some compression techniques [42].

Acknowledgements. This work is supported by Vision Semantics Limited, Alan Turing Institute Turing Fellowship, and Innovate UK Industrial Challenge Project on Developing and Commercialising Intelligent Video Analytics Solutions for Public Safety (98111-571149), Queen Mary University of London Principal's Scholarship.

References

[1] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2019.

[2] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *CVPR*, 2019.

[3] Yiming Chen, Shiji Song, Shuang Li, and Cheng Wu. A graph embedding framework for maximum mean discrepancy-based domain adaptation algorithms. *IEEE TIP*, 2019.

[4] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR-W*, 2020.

[5] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[6] Qi Dou, Daniel C Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. In *NeurIPS*, 2019.

[7] Hao-Zhe Feng, Zhaoyang You, Minghao Chen, Tianye Zhang, Minfeng Zhu, Fei Wu, Chao Wu, and Wei Chen. KD3A: Unsupervised multi-source decentralized domain adaptation via knowledge distillation. *arXiv preprint arXiv:2011.09757*, 2020.

[8] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[9] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. In *Proceedings of the Advances in Neural Information Processing Systems*, 2017.

[10] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.

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

[12] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 1998.

[14] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017.

[15] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *ICCV*, 2019.

[16] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *CVPR*, 2018.

[17] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020.

[18] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017.

[19] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[20] Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization. In *ICLR*, 2019.

[21] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[22] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *ICCV*, 2017.

[23] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS-W*, 2011.

[24] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *ECCV*, 2018.

[25] Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. Transferrable prototypical networks for unsupervised domain adaptation. In *CVPR*, 2019.

[26] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019.

[27] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. In *ICLR*, 2020.

[28] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

[29] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *ICCV*, 2019.

[30] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018.

[31] Seonguk Seo, Yumin Suh, Dongwan Kim, and Bohyung Han. Learning to optimize domain specific normalization for domain generalization. In *ECCV*, 2020.

[32] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *ICLR*, 2018.

[33] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR*, 2011.

[34] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

[35] Naveen Venkat, Jogendra Nath Kundu, Durgesh Kumar Singh, Ambareesh Revanur, et al. Your classifier can secretly suffice multi-source domain adaptation. In *NeurIPS*, 2020.

[36] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017.

[37] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *NeurIPS*, 2018.

[38] Hang Wang, Minghao Xu, Bingbing Ni, and Wenjun Zhang. Learning to combine: Knowledge aggregation for multi-source domain adaptation. In *ECCV*, 2020.

[39] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *ICLR*, 2020.

[40] Shujun Wang, Lequan Yu, Caizi Li, Chi-Wing Fu, and Pheng-Ann Heng. Learning from extrinsic and intrinsic supervisions for domain generalization. In *ECCV*, 2020.

[41] Guile Wu and Shaogang Gong. Decentralised learning from independent multi-domain labels for person re-identification. In *AAAI*, 2021.

[42] Hang Xu, Chen-Yu Ho, Ahmed M Abdelmoniem, Aritra Dutta, El Houcine Bergou, Konstantinos Karatsenidis, Marco Canini, and Panos Kalnis. Compressed communication for distributed deep learning: Survey and quantitative evaluation. Technical report, 2020.

[43] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *CVPR*, 2018.

[44] Luyu Yang, Yogesh Balaji, Ser-Nam Lim, and Abhinav Shrivastava. Curriculum manager for source selection in multi-source domain adaptation. In *ECCV*, 2020.

[45] Weichen Zhang, Wanli Ouyang, Wen Li, and Dong Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *CVPR*, 2018.

[46] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. 2018.

[47] Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. 2020.

[48] Sicheng Zhao, Guangzhi Wang, Shanghang Zhang, Yang Gu, Yaxian Li, Zhichao Song, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer. Multi-source distilling domain adaptation. In *AAAI*, 2020.

[49] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *AAAI*, 2020.

[50] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *ECCV*, 2020.

[51] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain adaptive ensemble learning. *arXiv preprint arXiv:2003.07325*, 2020.

[52] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *ICLR*, 2021.