Federated Multi-Target Domain Adaptation

Chun-Han Yao[1] Boqing Gong[2] Hang Qi[2] Yin Cui[2] Yukun Zhu[2] Ming-Hsuan Yang[1,2,3]

[1]UC Merced [2]Google [3]Yonsei University

Abstract

Federated learning methods enable us to train machine learning models on distributed user data while preserving its privacy. However, it is not always feasible to obtain high-quality supervisory signals from users, especially for vision tasks. Unlike typical federated settings with labeled client data, we consider a more practical scenario where the distributed client data is unlabeled, and a centralized labeled dataset is available on the server. We further take the server-client and inter-client domain shifts into account and pose a domain adaptation problem with one source (centralized server data) and multiple targets (distributed client data). Within this new Federated Multi-Target Domain Adaptation (FMTDA) task, we analyze the model performance of existing domain adaptation methods and propose an effective DualAdapt method to address the new challenges. Extensive experimental results on image classification and semantic segmentation tasks demonstrate that our method achieves high accuracy, incurs minimal communication cost, and requires low computational resources on client devices.

1. Introduction

Federated Learning (FL) [4, 28, 36, 27] aims to train a model using the data and computational resources on local devices to preserve privacy. While most existing FL methods assume labeled client data at our disposal, the assumption may be impractical for numerous computer vision tasks as it is difficult to obtain the ground-truth annotations. For instance, typical mobile phone users are unlikely to label object segmentation masks or bounding boxes on their abundant photos. Although many large-scale datasets for such tasks have been created and labeled by the vision community, storing and using them on local devices can cause memory, communication, or computational overload. Furthermore, there usually exists a large domain gap between the curated dataset and on-device user data.

To deal with the scenarios discussed above, we study a practical FL setting in this work. The server hosts a large-scale labeled dataset, viewed as the source domain, and each

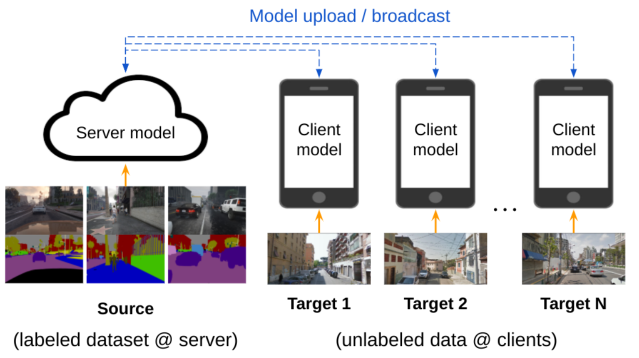


Figure 1. **Federated Multi-Target Domain Adaptation (FMTDA), a new problem setting this work studies.** The client models are trained in a distributed, privacy-preserving manner and aggregated on the server. Unlike typical FL scenarios, we aim to train a model using unlabeled client data and a labeled dataset available only on the server. The server and clients are data providers with distinct characteristics, resulting in both server-client and inter-client domain gaps. Therefore, it naturally forms a problem of single-source-multi-target domain adaptation.

client is considered a distinct target domain with unlabeled data. The client data has different underlying distributions due to various factors, *e.g.*, user habits, locations, and devices. The goal is to train a model that performs well for the clients by utilizing both the server data and distributed client data. We approach the server-client and inter-client domain mismatches by formulating the problem as federated multi-target domain adaptation (FMTDA), illustrated in Figure 1. Compared to typical FL or domain adaptation (DA), FMTDA introduces several new challenges. First, it prohibits the use of raw client data on the server. Most existing DA methods are not applicable to FMTDA since they directly access the target domain data by design. Second, the inter-client domain discrepancies increase the difficulty to perform FedAvg [26], a de facto algorithm in FL to aggregate client models on the server. Since each client model carries the idiosyncrasies of a distinct target domain, averaging the client models tends to cancel out or negatively affect the adaption efforts made on the client devices. Third, unlike the large-scale and representative target domain data in common DA frameworks, each client owns a limited amount of data. This implies that the adaptation methods cannot rely on a single client. Instead, they need to exploit the common properties underlying all client data while acknowledging individual data distribution. Finally, FMTDA should take into consideration the server-client communication overhead and the imbalanced computational resources. It is of great interest to develop an algorithm that requires low communication overhead and minimal computation on client devices. We highlight the aforementioned challenges by "federating" several popular DA methods [7, 25, 34], *i.e.*, modifying them so that they do not access the target domain data on the server end. Our analyses show that these methods do not account for the inter-client discrepancies, struggle with limited client data, and incur high computational cost on client devices.

To address the issues, we propose a *dual adaptation* (DualAdapt) approach which decouples the training framework into two parts: local adaptation on the client devices and global adaptation on the server. On a client device with low computational resources, we freeze the feature extractor in a deep neural network and learn a lightweight local classifier to capture the target characteristics. Meanwhile, we fit a parametric Gaussian mixture model (GMM) on the client data to encode its statistical distribution. The GMM parameters and local classifier respectively carry generative and discriminative information of the target domain, which are then uploaded to the server for the computationally heavy feature adaptation. On the server side, we jointly update the feature extractor and a global classifier after receiving the adapted local classifiers. Since the server cannot access client data, we design a proxy set to approximate the data distribution of target domains. We treat the source domain dataset as basis and apply a mixup [44] approach to construct a diverse and large-scale domain. The mixup set provides a wide support that covers the target domains, over which we can weight the instances by a GMM fitted on client data to approximate the corresponding target domain. To evaluate the model performance, we conduct extensive experiments on three image classification tasks and two semantic segmentation tasks. We compare our method with multiple centralized or federated DA baselines in terms of accuracy, communication overhead, and computational cost on client devices. The main contributions of this work are as follows:

- We introduce a new problem setting, FMTDA, for practical FL vision tasks. It deals with the domain gaps between the unlabeled, distributed client data and a labeled, centralized dataset on the server.
- We identify the key challenges in FMTDA and emphasize them by showing the degraded performance of prior DA methods as well as their "federated" version.
- We propose the DualAdapt approach to address the new challenges. The main idea is to self-train the local classifiers on client devices and adapt the heavy feature extractor on the server without accessing client data.

2. Related Work

Peng *et al.* [31] introduce a multi-source-single-target DA problem in the FL setting, which is closely related to our work but deals with an opposite scenario. They assume that the client data is labeled and the unlabeled server data is available on all local devices. With this setting, the whole model can be simultaneously trained on local devices but it poses heavy communication and computational demands on the clients. As such, this method cannot be applied to FMTDA for practical vision tasks. To the best of our knowledge, Peng *et al.* [31] are the first to study visual domain adaptation in FL, and our work is the first to formalize FMTDA with the consideration of memory, communication, and computational costs.

2.1. Unsupervised domain adaptation

Unsupervised Domain Adaptation (UDA) addresses the domain mismatch problem between labeled source data and unlabeled target data. Numerous UDA methods have been proposed to transfer the knowledge learned from a source domain to a target domain via centralized training. Existing approaches can be broadly categorized into divergence-based [2, 8, 39, 37, 25, 33, 3, 18], reconstruction-based [9, 40, 45, 19, 14], domain adversarial [7, 23, 38, 22], classifier discrepancy [34, 21], and self-training [45, 46] methods. In [2], Ben-David *et al.* introduce a divergence criteria to evaluate the domain shift and provide a generalization error bound for domain adaptation. Ghifary *et al.* [9] propose deep reconstruction-classification networks to address domain adaptation by learning an additional reconstruction task. Based on adversarial learning, Ganin and Lempitsky [7] design a gradient reversal layer to train a domain discriminator, which is widely used in the domain adaptation literature. Saito *et al.* [34] propose the MCD method to align source and target features by maximizing the discrepancy of two classifiers. Another group of methods utilize a model trained on source data to generate pseudo label on the target data in a self-training (ST) manner. Instead of assuming that the target data comes from a single domain, Yu *et al.* [42] and Gholami *et al.* [10] address a challenging multi-target UDA problem. However, these methods are not directly applicable to the FMTDA setting since they assume centralized data on a server. In addition, reconstruction-based and domain adversarial approaches require additional network modules such as domain discriminators or encoder-decoder architectures, which increase computational cost if applied to the client devices.

2.2. Federated learning

Federated learning [4, 28, 36, 27] is proposed to train a model in a distributed and privacy-preserving manner. The decentralized learning approaches enable multiple clients to collaboratively learn a model while keeping the training data on local devices. GiladBachrach *et al.* [11] propose CryptoNets to improve FL performance by enhancing the efficiency of data encryption. In [4], Bonawitz *et al.* introduce a secure aggregation method to update the machine learning models. Mohassel and Zhang [28] propose SecureML to support privacy-preserving collaborative training in a multi-client FL system. These methods mainly aim to learn a single global model that works well on centralized data within a single domain. Recently, several methods are proposed to address the non-*i.i.d* distribution of client data. Smith *et al.* [36] introduce federated multi-task learning, which learns a separate model for each node. In [24], Liu *et al.* propose semi-supervised transfer learning in a privacy-preserving setting. Hsu *et al.* [15, 16] analyze the effect of non-*i.i.d* client data in classification tasks and propose to improve the federated aggregation methods. On the other hand, Yu *et al.* [43] focus on the model performance on individual client data and train personalized models via local adaptation. Nonetheless, the models discussed above require full or semi-supervision on the client data. In [17], Jeong *et al.* deal with a disjoint FL scenario where server data is labeled and client data is partially labeled or unlabeled, but the server-client and inter-client domain gaps are not addressed.

3. Approach

Given a centralized, labeled dataset on the server as source domain D_S and the unlabeled data from multiple clients as target domains $({D_T}^i, \ldots, {D_T}^N)$, our goal is to train a model that performs well on the target data. Unlike centralized DA, the server cannot access the client data due to privacy concerns. We further prohibit the use of server dataset on client devices considering the limitation of local storage, computation, and communication. These constraints lead to a multi-party learning framework where all participants keep their data private and only interact via model parameters. While the server and client models share the same input and output space, their data exhibit domain discrepancies due to various client characteristics.

3.1. Preliminaries

In order to minimize the computational cost on client devices, we do not consider resource-consuming DA modules such as adversarial domain discriminators [7, 23, 38, 22] or encoder-decoder architectures [9, 40, 45, 19, 14]. Instead, we exploit maximum classifier discrepancy (MCD) [34], which only requires one additional classifier for domain alignment. A deep neural network classifier can be viewed as a classification head F stacking on top of a feature extractor G. MCD introduces a second classifier to identify the target data in D_T that is excluded by the support of source domain D_S. The objective of the classifiers F_1, F_2 can be expressed as:

$$\min_{F_1, F_2} \quad \mathcal{L}_{\text{ce}}(D_S) - \mathcal{L}_{\text{adv}}(D_T), \tag{1}$$

where $\mathcal{L}_{\text{ce}}(D_S)$ is a cross-entropy loss for classifying the labelded source data, and $\mathcal{L}_{\text{adv}}(D_T) = \mathbb{E}_{\boldsymbol{x}\sim D_T} \|F_1(G(\boldsymbol{x})) - F_2(G(\boldsymbol{x}))\|_1$ is the discrepancy between the classifier predictions over target data. Next, the feature extractor G is updated to align the classifier discrepancy as:

$$\min_{G} \quad \mathcal{L}_{\text{adv}}(D_T). \tag{2}$$

These training steps are applied alternately and repeated until convergence.

3.2. Dual adaptation for FMTDA

The decoupling of classifier and feature extractor in MCD makes it a natural fit for FMTDA. Since it is relatively lightweight to adapt a classifier to the target domain (Eq. 1), we assign it to the client devices and update the feature extractor G on the server (Eq. 2). In addition to the computational efficiency, this design also effectively reduces the communication overhead. The server broadcasts the whole model to the clients, while the clients only need to upload the classifiers to the server. Nonetheless, in this vanilla MCD method, the clients need access to the source data to update F_1, F_2 (Eq. 1) and the server requires the target data to adapt G (Eq. 2). Neither of the requirements satisfies the FMTDA constraints. More importantly, each client in FMTDA observes a distinct source-target domain pair due to the non-*i.i.d.* data of different clients. Simply averaging the feature extractors and classifiers on the server can easily lead to canceling effect or negative transfer. We address the issue by designating a unique local classifier ${F_l}^i$ to each client i. Meanwhile, we maintain a global classifier F_g to account for the discriminative properties shared by the source and all target domains. To tackle the data privacy constraints, we propose a self-training approach on the client end and a proxy target domain on the server. The DualAdapt framework is illustrated in Figure 2.

3.2.1 Self-training local classifiers on the clients.

In Eq. 1, the term $\mathcal{L}_{\text{ce}}(D_S)$ is meant to preserve the discriminative power of classifiers and avoid trivial solutions caused by the discrepancy term $\mathcal{L}_{\text{adv}}(D_T)$. However, it is infeasible to update the local classifiers $\{{F_l}^i\}$ on client devices since minimizing $\mathcal{L}_{\text{ce}}(D_S)$ requires the source domain data. We propose the following variations to self-train classifiers without copying the source data to client devices. First, we freeze the global classifier F_g and only adapt the local classifier ${F_l}^i$ for client i. As a result, the domain discrepancy loss for client i only applies to the local classifier ${F_l}^i$, which can be written as:

$$\mathcal{L}^i_{\text{adv}}(\boldsymbol{x}) = \|F_g(G(\boldsymbol{x})) - {F_l}^i(G(\boldsymbol{x}))\|_1, \tag{3}$$

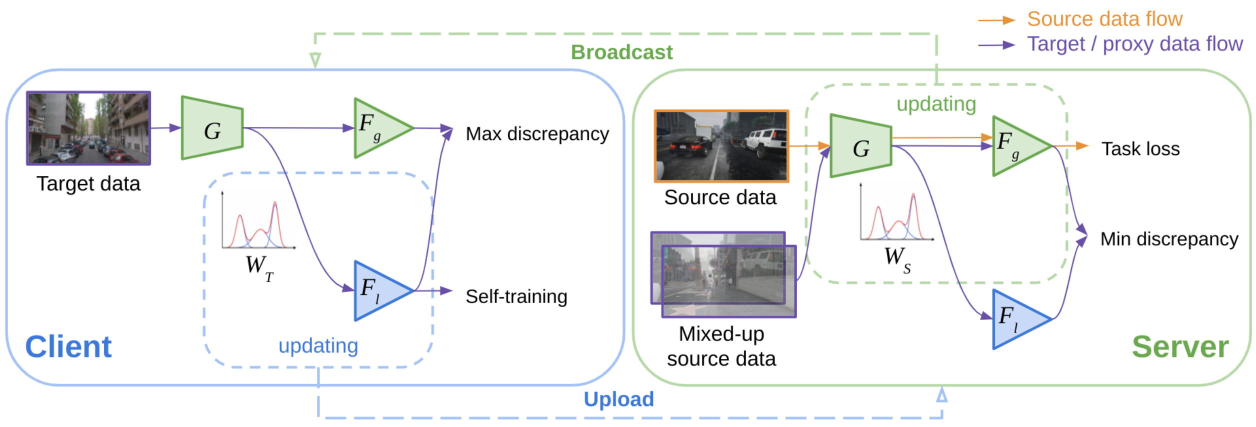


Figure 2. **DualAdapt framework overview** (best viewed in color). Our model consists of a feature extractor G, a global classifier F_g, and a local classifier F_l for each client. G and F_g are updated on the server and broadcast to the clients. Each client then updates its F_l during local adaptation. We also fit a GMM model on both the target (W_T) and source (W_S) data features to weight the examples for training.

It not only reduces the communication and local computational costs but stabilizes the local adaptation process. Second, we preserve the discriminativeness of local classifiers via a self-training loss $\mathcal{L}_{\text{st}}$. It is defined as the cross-entropy between the prediction of the local classifier ${F_l}^i$ and the pseudo label generated by the global classifier F_g. The pseudo label $\hat{y}$ of class c of input $\boldsymbol{x}$ is given as:

$$\hat{y}_c(\boldsymbol{x}) = \begin{cases} 1, & \arg\max_k [F_g(G(\boldsymbol{x}))]_k = c \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

where $[\boldsymbol{p}]_k$ is the k-th element of a probability vector $\boldsymbol{p}$.

Since the prediction of F_g may not be perfect due to domain mismatch, we use a GMM model fitted on the source data to weight the target examples. Intuitively, we give more confidence to the prediction of F_g if the example lies closer to the source distribution. Thus, we send the source GMM parameters W_S from the server to each client. For each example $\boldsymbol{x}$ in the target domain, we calculate the GMM probability $W_S(\boldsymbol{x})$ as the confidence of F_g and weight the self-training loss accordingly. The local optimization objective for client i can be expressed as:

$$\min_{{F_l}^i} \ \mathbb{E}_{\boldsymbol{x}\sim {D_T}^i} - \mathcal{L}^i_{\text{adv}}(\boldsymbol{x}) + \ \lambda_{\text{st}} \ W_S(\boldsymbol{x}) \ \mathcal{L}^i_{\text{st}}(\boldsymbol{x}), \tag{5}$$

where λ_{st} is a weighting hyper-parameter. Note that the feature extractor G and global classifier F_g are both fixed on the client devices. Therefore, the local computation and uploading costs are lightweight as both involve the local classifier ${F_l}^i$ only. In addition to the local classifier that captures the discriminative properties of client data, we fit a GMM to describe its generative statistics, thereby allowing the server to construct a proxy set to the client data. After local adaptation, each client uploads its local classifier F_l^i and GMM parameters ${W_T}^i$ to the server.

3.2.2 Feature alignment via mixup on the server.

Ideally, the server gathers the local classifier and data from the clients and adapts the feature extractor G as described in Eq. 2. However, the server cannot access target data in the FL settings. To align features from two domains, we need a certain proxy to approximate the target distribution using only the server data. In addition to widely-used image transformations, e.g., flipping, cropping, and color jittering, we construct a proxy of the target domains by re-weighing the mixup [44] of abundantly available server data. Mixup is developed to regularize neural network training by densely sampling the convex combinations of training examples. The convex hull of large-scale source data likely overlaps with the support of target domains considerably. Moreover, the relationship between empirical risk minimization and mixup is derived [44] and empirically shown to effectively improve model generalization, robustness to adversarial examples, and training stability. These properties are of particular importance in FMTDA since decentralized training with non-*i.i.d.* and unlabeled client data tends to be unstable. By fitting a GMM on each target domain, we can further sample a proxy set according to the data density of each client. Note that GMM encodes global statistics in the feature space, which carry less private information than the parameters/gradients of a client model.

Specifically, we randomly average two source instances $\boldsymbol{x}_m, \boldsymbol{x}_n \sim D_S$ in a data batch as a mixup instance $\boldsymbol{x}_{mn} = (\boldsymbol{x}_m + \boldsymbol{x}_n)/2$. Given the GMM parameters ${W_T}^i$ from client i, the server uses it to weight each mixup example $\boldsymbol{x}_{mn}$, denoted by ${W_T}^i(\boldsymbol{x}_{mn})$. The objective for the server-side adaptation is a weighted average over the mixup examples:

$$\min_G \sum_{i=1}^N \mathbb{E}_{\boldsymbol{x}\sim\text{mixup}(D_S)} {W_T}^i(\boldsymbol{x}) \ \mathcal{L}^i_{\text{adv}}(\boldsymbol{x}), \tag{6}$$

Algorithm 1 DualAdapt

```
Input: Source domain $D_S = \{(\boldsymbol{x_s}, y_s)\}$, target domains $({D_T}^1, ..., {D_T}^N) = (\{\boldsymbol{x_t}^1\}, ..., \{\boldsymbol{x_t}^N\})$, Number of client iterations $R_c$, Number of server iterations $R_s$
Output: Feature extractor $G$, global classifier $F_g$, local classifiers $({F_l}^1, ..., {F_l}^N)$
 1: Pre-train $G$ and $F_g$ on server with cross-entropy loss
 2: repeat
 3:     Fit global GMM $W_S$ on source data
 4:     Broadcast server models $G, F_g, W_S$ to each client
 5:     # Local adaptation for client $i$ :
 6:     Initialize local classifier ${F_l}^i \leftarrow F_g$
 7:     Initialize local GMM ${W_T}^i$
 8:     for $r = 1 : R_c$ do
 9:         Sample mini-batch ${\boldsymbol{x_t}}^i$ from ${D_T}^i$
10:         Update ${F_l}^i$ on ${\boldsymbol{x_t}}^i$ with Eq. 5
11:         Update ${W_T}^i$ on ${\boldsymbol{x_t}}^i$
12:     Upload ${F_l}^i$ and ${W_T}^i$ to the server
13:     # Server optimization:
14:     for $r = 1 : R_s$ do
15:         Sample mini-batch $(\boldsymbol{x_s}, y_s)$ from $D_S$
16:         Generate mixup data $\boldsymbol{x_{s'}}$ from $\boldsymbol{x_s}$
17:         Update $G$ on $\boldsymbol{x_{s'}}$ with Eq. 6
18:         Fine-tune $G$ and $F_g$ on $(\boldsymbol{x_s}, y_s)$ with cross-entropy loss
19: until convergence
```

where N is the total number of clients. The weighting mechanism allows the feature extractor G to focus on the mixup instances that are closer to a target domain when adapting the extracted features.

3.3. Model training and inference

Algorithm 1 describes the algorithmic details of the DualAdapt method and Figure 2 illustrates the main steps. We first pre-train the server model (G, F_g, W_S) on the labeled source data D_S. Then, the model is broadcast to all clients and each local classifier ${F_l}^i$ is initialized by F_g. After local optimization (Eq. 5), each client uploads its local classifier and GMM parameters W_T to the server. The server first updates the feature extractor and global classifier (Eq. 6) and then fine-tune the model (G, F_g) using a cross-entropy loss over the source data before broadcasting them back to the clients. When fitting the GMM models, we reduce the feature dimension with PCA to preserve at least 80% of the original energy and empirically choose the number of mixture components as twice the number of classes. The client models are trained with mini-batch gradient descent and the server model is updated by a momentum optimizer to stabilize the training. In the inference phase, we ensemble the predictions of global and local classifiers as:

$$\tilde{y}(\boldsymbol{x}) = (F_g(G(\boldsymbol{x})) + {F_l}^i(G(\boldsymbol{x})))/2, \tag{7}$$

where $\boldsymbol{x}$ is a test example from client i.

4. Experiments and Analyses

Datasets. We experiment with three FMTDA tasks: digit classification on the Digit-Five dataset [31], image classification on the DomainNet dataset [30], and semantic segmentation with adaptation from the synthetic GTA5 dataset [32] to the real-world CrossCity dataset [6]. Figure 3 shows some sample images from these datasets. Digit-Five is composed of five benchmarks for digit recognition: MNIST [20], Synthetic Digits [7], MNIST-M [7], SVHN [29], and USPS. The DomainNet dataset contains 596K images of 345 classes from 6 modalities: clipart, inforgraph, painting, quickdraw, real, and sketch. Each modality is viewed as a domain in this paper. The GTA5 dataset contains 25k street-view images simulated from computer games. It provides dense pixel-wise labels for semantic segmentation with 19 classes. The CrossCity dataset consists of real-world street scenes collected from four different cities: Rio, Rome, Taipei, and Tokyo. There are 3.2k training images and 100 testing images for each city. Since the CrossCity dataset is labeled with only 13 classes, we train and evaluate the models using the overlapped 13 classes between GTA5 and CrossCity. To simulate a practical scenario, we assume that each client (target domain) only possesses a limited amount of data. The main results presented here use 10% data from each target domain in the large-scale Digit-Five and DomainNet datasets, and the supplementary material contains results of other settings.

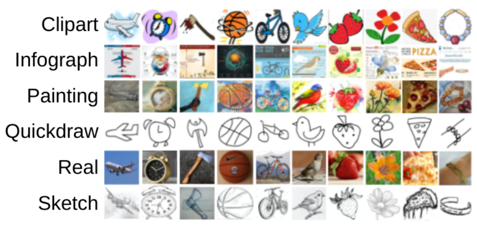
Baselines and upper bounds. Since FMTDA is new in the literature, we mainly evaluate our approach, DualAdapt, with the following methods: 1) centralized models trained on the source data only, 2) centralized MCD [34] (Cent-MCD) using centralized training in a multi-target DA setting, 3) federated versions of several DA methods (Fed-DAN [25], Fed-DANN [7], Fed-MCD [34]), and 4) federated oracle (Fed-oracle), which assumes that DualAdapt can access the target data on the server.

Implementation details. We use the standard FL protocol to extend the centralized DA methods to our setting, i.e., updating models locally and aggregating them on the server using the Federated Averaging (FedAvg) algorithm. Note that the federated DA baselines require a copy of the source dataset for every client, which gives these schemes some advantage over our method. The federated oracle model replaces the mixup data on the server with the target domain examples and discards the GMM weighting mechanism. We implement our model using TensorFlow [1]. The supplementary material contains more implementation details, model architecture, and additional experimental results on two other adaptation tasks: image classification on the Office-Caltech10 dataset [12] and semantic segmentation on the BDD100k dataset [41].

4.1. Task accuracy

Digit-Five. We use the MNIST dataset as the source domain and the rest as the target domains (i.e., one-to-four adaptation task). Our models include 4 convolutional lay-

(a) Digit-Five dataset



(b) DomainNet dataset

(c) CrossCity dataset

Figure 3. **We evaluate the model performance on three different datasets: Digit-Five, DomainNet, and CrossCity.** First, we adapt from MNIST to four other datasets for digit classification. Second, we take turn adapting from one image modality in DomainNet to the rest. Finally, we perform cross-city adaptation in semantic segmentation from the synthetic GTA5 dataset to real-world street images.

Table 1. **Quantitative evaluations on the Digit-Five dataset.** We report the classification accuracy (%) on the target domains. The MCD method produces a clear accuracy gain from the source-only baseline but does not perform well in the federated setting. Our method improves the performance significantly compared to the federated baselines.

Method	MNIST-M	SVHN	Synthetic	USPS	Avg
Source only	26.1	10.4	26.9	57.7	30.3
Cent-MCD [34]	28.5	12.3	29.2	70.8	35.2
Fed-oracle	28.1	12.0	28.3	69.5	34.5
Fed-DAN [25]	26.6	10.4	27.1	59.4	30.9
Fed-DANN [7]	26.9	10.6	27.8	59.9	31.3
Fed-MCD [34]	27.2	10.7	27.4	61.6	31.7
DualAdapt (ours)	**27.7**	**11.9**	**28.0**	**68.9**	**34.1**

ers for feature extraction and two fully-connected layers for each classifier. Table 1 reports the classification accuracy on the target domains and Figure 4 shows the results with various amount of target data for training. Compared to the source-only baseline, Cent-MCD improves the classification accuracy considerably (from 30.3% to 35.2%), confirming the efficacy of domain adaptation. Both Cent-MCD and Fed-oracle access the client data on the server, which serve as the upper bounds of our method. We emphasize that DualAdapt can almost match the performance of Fed-oracle (34.1% vs. 34.5% on average). In the FMTDA setting, DualAdapt alleviates the inter-client and source-target domain mismatches. While the other federated DA methods additionally have access to the source data on client devices, they only tackle the source-target domain discrepancy. The results show that DualAdapt achieves much higher performance (34.1%) than the federated DA methods (30.9%, 31.3%, and 31.7%), demonstrating the necessity of modeling the inter-client domain gaps.

DomainNet. We perform cross-modality adaptation on the DomainNet dataset [30] using ResNet101 [13] for feature extraction. Each modality in the DomainNet is taken as the source domain in turn. The quantitative results are shown in Table 2. The performance of evaluated methods are similar to that in the Digit-Five dataset. Adapting from "quick-

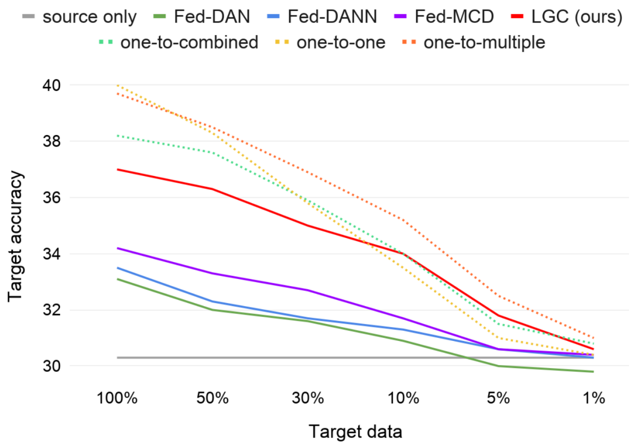


Figure 4. **Quantitative evaluations of different DA methods on the Digit-Five dataset.** The solid lines show the federated DA methods and the dash lines are the centralized MCD [34] method under different training scenarios. We evaluate the models trained with various amount of data per client. Our model achieves consistently higher accuracy compared to the federated baselines, and can perform better than several centralized methods when the target data on each client device is limited.

draw" and "sketch" to the rest are more challenging due to their larger modality gaps than other domains. However, DualAdapt is able to achieve consistent performance gains over the other federated DA methods.

GTA5 to CrossCity. For this semantic segmentation task, the domain adaptation is from the synthetic GTA5 dataset [32] to the four real-world cities in CrossCity [6]. We use the MobileNetv2 [35] with multiplier $\alpha = 0.5$ as backbone and DeepLabv3 [5] without decoder as our segmentation model. Although GTA5 is a large-scale dataset, the diverse scene structures across different target cities pose a great challenge for adaptation. Table 3 shows the quantitative results of the evaluated methods based on the mean intersection-over-union (mIoU) metric. While Cent-MCD achieves clear performance gain over the source-only baseline in the centralized setting, the federated adaptation methods do not preform as well in the FL setting. Our

Table 2. **Quantitative evaluations on the DomainNet dataset.** We take turn using one domain as source and the rest as targets and report the average accuracy (%) on the target domains. Our method (DualAdapt) achieves similar accuracy to the centralized and federated upper bounds, which is significantly higher than the federated baselines.

Method	Clipart →	Infograph →	Painting →	Quickdraw →	Real →	Sketch →	Avg
Source only	25.7	19.3	29.6	4.4	31.5	10.5	20.2
Cent-MCD [34]	29.6	23.4	34.1	7.2	36.1	14.2	24.1
Fed-Oracle	29.2	22.9	33.8	6.6	35.8	13.5	23.6
Fed-DAN [25]	25.9	19.7	30.3	4.6	31.1	11.4	20.5
Fed-DANN [7]	25.5	20.1	30.4	4.7	31.3	10.8	20.5
Fed-MCD [34]	27.1	19.9	31.4	4.7	32.5	10.4	21.0
DualAdapt (ours)	**29.0**	**22.3**	**33.5**	**6.0**	**35.7**	**13.2**	**23.3**

Table 3. **Quantitative evaluations on GTA5-to-CrossCity.** In this challenging segmentation task, the federated baselines produce minimal mIoU gain on the target domains whereas our method improves the performance by a clear margin.

Method	Rio	Rome	Taipei	Tokyo	Avg
Source only	27.9	27.6	26.0	28.2	27.4
Cent-MCD [34]	31.1	30.6	28.8	31.6	30.5
Fed-oracle	30.3	28.4	28.3	30.7	29.4
Fed-DAN [25]	27.3	26.4	26.0	28.5	27.1
Fed-DANN [7]	28.6	26.0	26.6	28.6	27.5
Fed-MCD [34]	27.7	27.3	26.5	29.0	27.6
DualAdapt (ours)	**29.2**	**28.0**	**27.6**	**30.7**	**28.9**

method improves from the baseline by 1.3 mIoU, approaching the oracle upper bound (28.9 vs. 29.4).

4.2. Communication and computational costs

In addition to measuring the task accuracy on target data, we also evaluate the server-client communication and on-device computational costs. For communication, we calculate the number of model parameters that need to be transmitted in two directions: upload (client to server) and broadcast (server to client). Regarding the computational cost, we emphasize the importance of low training cost on the client devices since they are usually equipped with limited computational resources. Specifically, we estimate the number of floating-point operations (FLOPs) incurred by a data example in a forward pass and backpropagation during training. Without loss of generality, we divide most methods into a feature extractor G, a classifier F, and optionally a domain discriminator D. For each module $m \in \{G, F, D\}$, we denote by $|m|$ the number of its parameters and by $\|m\|$ the FLOPs of a forward pass of one data example. The FLOPs in backpropagation are typically similar to the forward pass FLOPs $\|m\|$ in our experiments, so we estimate the on-device training cost by $2\|m\|$ if the module is trainable.

We use the federated DA baselines and our method to demonstrate how to calculate the communication and on-device computational costs. Recall that in the federated DA baselines, each client has access to its own target domain data as well as the source data from server.

- The DANN method [7] aligns the source and target domains by inserting a gradient reversal layer between the feature extractor and domain discriminator. We train a domain discriminator for each client and update it locally with the main network. During local optimization, each target domain example is passed through G and D, and source domain example is passed through all the modules (G, F, and D). Hence, the on-device computational cost is $2(\|G\| + \|D\|) + 2(\|G\| + \|F\| + \|D\|)$ in Fed-DANN. For data communication, each client uploads its G and F to the server, and the server broadcasts the aggregated G and F back to the clients. The communication overhead is $|G| + |F|$ in both directions.
- Fed-MCD [34] trains a feature extractor and two classifiers for each federated client. The computational cost is thus $2(\|G\| + 2\|F\|) + 2(\|G\| + 2\|F\|)$, and the communication overhead is $|G| + 2|F|$.
- In DualAdapt, the feature extractor and global classifier are fixed on client devices, and we only pass the target data through the local model. It reduces the computational cost to $\|G\| + 3\|F\|$, including one forward pass over G, F_g, and F_l and a backpropagation pass over F_l. For data communication, we upload $|F|+|W|$ parameters and broadcast $|G| + |F| + |W|$ parameters.

Generally, the communication and computational costs are dominated by the feature extractor G of a model. DualAdapt reduces the computational and upload costs considerably compared to the federated baselines as we do not update G on-device and do not pass the source data through the local model. Fitting GMM models to the target data barely creates additional costs per data example per iteration since the feature dimension is reduced. In Table 4, we show the communication and computational costs incurred by our method and other federated baselines. In all three experiments, DualAdapt requires approximately 1/4 FLOPS of other methods during the client-end training. Moreover, the number of model parameters that need to be uploaded to the server is reduced considerably. The results demonstrate that DualAdapt can perform the adaptation efficiently with the minimal costs of communication and computation.

Table 4. **Evaluations of communication and computational costs.** We calculate the client-end FLOPS as computational cost during training and number of model parameters to upload + broadcast as communication overhead. Our method (DualAdapt) reduces the communication and computational costs significantly compared to the other baselines.

	Digit-Five		**DomainNet**		**GTA5-to-CrossCity**	
Method	Computation	Communication	Computation	Communication	Computation	Communication
Fed-DAN [25]	314.0M	493K + 493K	36.1B	2.5M + 2.5M	31.5B	45.4M + 45.4M
Fed-DANN [7]	314.3M	493K + 493K	36.1B	2.5M + 2.5M	32.0B	45.4M + 45.4M
Fed-MCD [34]	314.6M	510K + 510K	36.1B	2.5M + 2.5M	32.6B	46.1M + 46.1M
DualAdapt (ours)	**78.7M**	**18K + 494K**	**9.0B**	**0.02M + 2.5M**	**8.4B**	**1.6M + 46.2M**

Table 5. **Ablative evaluations on the Digit-Five dataset.** We show that DualAdapt significantly reduces the communication and computational costs by decoupling client and server model training. The individual components of our framework like ST and GMM improve the accuracy and barely require additional communication and computational costs.

Method	Client	Server	Accuracy (%)	Computation (FLOPS)	Communication (# parameters)
Fed-MCD [34]	MCD target	-	31.7	314.6M	510K + 510K
DualAdapt	MCD target	MCD mixup	32.8	**78.5M**	**17K + 493K**
DualAdapt	MCD target + ST	MCD mixup	33.5	**78.5M**	**17K + 493K**
DualAdapt	MCD target + ST	MCD mixup + GMM	34.1	78.7M	18K + 494K
Fed-oracle	MCD target + ST	MCD target	34.5	78.5M	17K + 493K

4.3. Ablation studies

To evaluate the effectiveness and costs of individual components in our framework, we perform ablation studies on the Digit-Five experiment. Table 5 shows, by introducing local and global classifiers, we achieve a 1.1% accuracy gain and require only a quarter of the computational cost and half of the communication overhead compared to Fed-MCD. Self-training (ST) further improves the accuracy by 0.7% and barely increases the computation overhead. With the GMM weighting mechanism, the accuracy of our full model is close to the oracle while having only 1K additional parameters to upload and broadcast. It demonstrates the effectiveness of weighting mixup data as target proxy.

5. Centralized multi-target domain adaptation

Our work is closely related to centralized multi-target DA. In a typical FL system, the client models are trained on locally collected data which is non-*i.i.d.* due to different user characteristics. The problem of domain mismatch among multiple clients, or target domains, is challenging even in a centralized training framework. Although some methods have been proposed for centralized multi-target DA [42, 10], the best way to adapt to diverse domains in real-world scenarios remains unclear. To disentangle the challenges of multi-target DA and distributed training, we also look into the problem in a centralized setting. The idea is to investigate how the performance of a DA method changes after eliminating the federated setting.

When dealing with multiple target domains, some common solutions include 1) treating the adaptation to each target as a separate DA task (one-to-one), 2) combining all target data into one domain and solving it as single-source-single-target DA (one-to-combined), and 3) adapting to multiple targets simultaneously by decomposing model parameters or feature representation into shared and private components (one-to-multiple). We evaluate MCD [34] with these settings using the Digit-Five experiment. In the one-to-multiple setting, the model is decomposed into a shared feature extractor and two classifiers for each target domain. As shown in Figure 4 (dash lines), the one-to-one adaptation models perform better when sufficient training data from the target domain is available. In a practical scenario with fewer target examples, the one-to-multiple method can more effectively adapt to the target domains. Note that the one-to-one adaptation requires multiple copies of the model parameters while one-to-combined needs only one. However, the one-to-combined method is not applicable in federated learning since the client data is private. The observations justify our one-to-multiple FL model design since the federated clients usually possess limited data.

6. Conclusion

Multi-target domain adaptation is a natural and challenging problem in federated learning. While most existing methods assume labeled and/or *i.i.d* client data, we consider a practical setting: multi-target unsupervised domain adaptation. We show that naively applying centralized DA methods on federated client devices not only leads to poor performance but also costs considerable communication and computational overheads. To address this, we propose a simple yet effective framework, DualAdapt, which requires minimal FL training costs. Extensive experiments on image classification and semantic segmentation demonstrate that DualAdapt can achieve significant performance gain compared to centralized and federated baselines. We hope that this work will encourage more research attention to this novel and crucial topic.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[3] Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *ECCV*, pages 447–463, 2018.

[4] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[5] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[6] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *ICCV*, pages 1992–2001, 2017.

[7] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015.

[8] Muhammad Ghifary, W Bastiaan Kleijn, and Mengjie Zhang. Domain adaptive neural networks for object recognition. In *Pacific Rim International Conference on Artificial Intelligence*, pages 898–904, 2014.

[9] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *ECCV*, pages 597–613, 2016.

[10] Behnam Gholami, Pritish Sahu, Ognjen Rudovic, Konstantinos Bousmalis, and Vladimir Pavlovic. Unsupervised multi-target domain adaptation: An information theoretic approach. *TIP*, 29:3993–4002, 2020.

[11] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*, pages 201–210, 2016.

[12] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pages 2066–2073, 2012.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[14] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, pages 1989–1998, 2018.

[15] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[16] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In *ECCV*, 2020.

[17] Wonyong Jeong, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. Federated semi-supervised learning with inter-client consistency. *arXiv preprint arXiv:2006.12097*, 2020.

[18] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *CVPR*, pages 4893–4902, 2019.

[19] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *ICCV*, pages 10285–10295, 2019.

[22] Alexander H Liu, Yen-Cheng Liu, Yu-Ying Yeh, and Yu-Chiang Frank Wang. A unified feature disentangler for multi-domain image translation and manipulation. In *NIPS*, pages 2590–2599, 2018.

[23] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NIPS*, pages 469–477, 2016.

[24] Yang Liu, Tianjian Chen, and Qiang Yang. Secure federated transfer learning. *arXiv preprint arXiv:1812.03337*, 2018.

[25] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, pages 2208–2217, 2017.

[26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[27] Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 35–52, 2018.

[28] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy*, pages 19–38, 2017.

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

[30] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, pages 1406–1415, 2019.

[31] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*, 2019.

[32] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, pages 102–118, 2016.

[33] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *PAMI*, 41(4):801–814, 2018.

[34] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, pages 3723–3732, 2018.

[35] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[36] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *NIPS*, pages 4424–4434, 2017.

[37] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, pages 443–450, 2016.

[38] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, pages 7167–7176, 2017.

[39] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[40] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*, pages 2849–2857, 2017.

[41] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, pages 2636–2645, 2020.

[42] Huanhuan Yu, Menglei Hu, and Songcan Chen. Multi-target unsupervised domain adaptation without exactly shared categories. *arXiv preprint arXiv:1809.00852*, 2018.

[43] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020.

[44] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[45] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017.

[46] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. In *ICCV*, pages 5982–5991, 2019.