Assisted Unsupervised Domain Adaptation

Cheng Chen
Department of ECE
University of Utah
Salt Lake City, UT 84112
u0952128@utah.edu

Jiawei Zhang
School of Statistics
University of Minnesota
Minneapolis, MN 55455
zhan4362@umn.edu

Jie Ding
School of Statistics
University of Minnesota
Minneapolis, MN 55455
dingj@umn.edu

Yi Zhou
Department of ECE
University of Utah
Salt Lake City, UT 84112
yi.zhou@utah.edu

***Abstract*—Unsupervised domain adaptation (UDA) is a popular machine learning technique that allows one to train models over diverse data collected from different domains. However, this technique requires the learner to collect a large number of properly labeled data samples, which can be costly and unrealistic in many applications. In this work, we propose a decentralized assisted learning framework for UDA. In this framework, a learner has only a limited number of labeled data samples collected from a certain source domain and aims to train a classifier for the target domain. To improve domain adaptation performance, it seeks assistance by interacting with an external service provider, who possesses many labeled data samples collected from a related source domain. We develop an assisted UDA algorithm that avoids data sharing and can significantly improve the learner's domain adaptation performance within a few rounds of interaction. Experiments using deep neural networks on benchmark datasets demonstrate the effectiveness of this algorithm.**

I. Introduction

The effectiveness of modern machine learning is highly dependent on the availability of a large, high-quality set of labeled data samples, meaning that the data is free of noise and its distribution is closely aligned with that of the test data samples [1], [2]. However, these requirements are often difficult to meet in practice due to two main factors: (i) collecting and annotating such a large amount of data is often costly and time-consuming, so one usually only has access to a limited number of labeled samples; (ii) there is often a significant distribution shift between training and test samples, leading to high generalization error for the learning algorithms.

To address these issues, *unsupervised domain adaptation (UDA)* has become a popular technique that has been extensively studied in the literature. For example, [3] proposed an adversarial model for UDA. This model consists of a feature extractor, a label predictor, and a domain classifier. By minimizing the label prediction loss and maximizing the domain classification loss simultaneously, the model can learn transferable features from the labeled source data that can be adapted to the unlabeled target data (i.e., the test samples) and achieve domain-invariant learning. Additionally, other works have developed multi-source UDA approaches [4]–[8], in which the goal is to learn features that are transferable to the unlabeled target data when labeled samples are available from multiple source domains. Please refer to Fig. 1 for an illustration of UDA-type of models.

However, ensuring good performance in unsupervised domain adaptation often requires access to a large, centralized set of labeled data samples from various source domains, which can be unachievable in practical application scenarios. In the era of big data, the proliferation of data collection methods and machine learning techniques have brought opportunities for modeling with external assistance [9], [10]. The UDA model with poor performance may improve significantly with assistance from a service provider with an external dataset. The challenges of this learning scenario include the followings: 1) the external dataset may have a different distribution from local datasets; 2) assistance from a service provider may be costly due to high service fees charged for frequent exchanges of information and updates; 3) decentralized learning algorithms are required due to the absence of a central server.

A. Main Contribution

To address the above challenges, we propose a *decentralized assisted learning framework for UDA*. In this framework, we consider a scenario where the learner has a limited number of labeled samples collected from a single-source domain and aims to use these labeled samples to train a classifier on a different target domain. We assume that an external service provider, such as a data company or organization, may possess labeled samples from a related but different source domain. In this case, the learner can seek assistance from the service provider to improve domain adaptation performance without data sharing, through a few rounds of communication. We present an assisted UDA algorithm and evaluate it using deep neural network models on benchmark datasets, demonstrating its effectiveness.

B. Other Related Work

Many distributed or decentralized methods have gained popularity in recent years to improve the machine learning performance of a learner with limited data and computation resources. Two such methods are assisted learning and federated learning, whose related work are summarized below.

Assisted Learning (AL): AL [10]–[12] is a decentralized learning framework that allows organizations to autonomously improve their learning quality within only a few assistance rounds. Previous research on AL has focused on vertically partitioned data, in which entities hold data with different feature variables collected from the same cohort of subjects. In

contrast, this work considers horizontally partitioned data, in which clients hold local data with the same feature variables but from different subjects. Our setting is therefore closely aligned with a recent AL framework developed in [13] for assisting organizations with horizontally partitioned data, which demonstrated that a learner organization (e.g., small labs) could significantly and steadily improve its learning performance through interaction with an external service provider (e.g., large institutes) within a few rounds. In this work, we will develop an AL approach for a learner performing domain adaptation learning to receive assistance from a provider to leverage multiple source domains.

Federated Learning (FL): FL [14]–[17] is a distributed learning framework that allows for the joint learning of a model through the averaging of locally learned model parameters without the need for training data to be transmitted. It utilizes the resources of large numbers of edge devices, also known as "clients," to achieve a global objective coordinated by a central server. Vertical Federated Learning methods involve splitting sub-networks for local clients to jointly optimize a global model [18]–[21], but require frequent batchwise synchronization of backward gradients to converge [21].

II. Assisted Unsupervised Domain Adaptation

In this section, we introduce our assisted unsupervised domain adaptation framework. We first present the problem formulation, review the existing centralized multi-source domain adaptation approach, and then present our proposed assisted domain adaptation algorithm.

A. Problem Formulation

We consider a learning scenario in which a learner (denoted as L) seeks to train a classification model for predicting unlabeled test sample $\mathcal{D}(\mathrm{T})$ collected from a target domain T. However, the learner only has a limited number of labeled data $\mathcal{D}(\mathrm{S})$ collected from a related source domain S. Using these labeled data for direct domain adaptation may result in poor performance. To improve domain adaptation performance, the learner L seeks assistance from an external service provider (denoted as P), who possesses a large number of labeled data $\mathcal{D}(\mathrm{S}')$ collected from a source domain S' closely related to the learner's domains S and T. As L needs to purchase service provided by P, it is desirable for their interactions to follow the assisted learning protocols outlined in [13], which specify that (i) no data sharing should occur and (ii) learner should achieve a high-performance gain within a few interaction rounds. It is also assumed that both L and P have sufficient computational resources.

Centralized Multi-Source UDA. Consider an *ideal* case where L has access to P's data. A simple solution is to apply the standard multi-source UDA approach with centralized data. Here, we briefly review this approach, which serves as the basis for developing our assisted UDA algorithm later.

Specifically, given labeled data samples $\mathcal{D}(\mathrm{S}), \mathcal{D}(\mathrm{S}')$ from two different source domains, multi-source UDA aims to learn invariant features that are transferable to the unlabeled target data samples $\mathcal{D}(\mathrm{T})$. In [8], the authors proposed a multi-source DA model, as shown in Fig. 1, which consists of a feature extractor θ_f, a label predictor θ_y, and two domain classifiers θ_{d1}, θ_{d2}. The learning objective function consists of a label classification loss $\mathcal{L}(\theta_f, \theta_y; \mathcal{D}(\mathrm{S}, \mathrm{S}'))$ that aims to train a label prediction model $\{\theta_f, \theta_y\}$ using the labeled source data samples $\mathcal{D}(\mathrm{S}, \mathrm{S}') := \mathcal{D}(\mathrm{S}) \cup \mathcal{D}(\mathrm{S}')$, and a domain classification loss $\mathcal{L}(\theta_f, \theta_{d1}; \mathcal{D}(\mathrm{S}, \mathrm{T})) + \mathcal{L}(\theta_f, \theta_{d2}; \mathcal{D}(\mathrm{S}', \mathrm{T}))$ that aims to extract domain-invariant features between domains (S, T) and $(\mathrm{S}', \mathrm{T})$ by letting the feature extractor θ_f compete with the domain classifiers θ_{d1}, θ_{d2}. Specifically, the optimization aims to *minimize* the label classification loss over $\{\theta_f, \theta_y\}$, and *minimize* (respectively *maximize*) the domain classification loss over $\{\theta_{d1}, \theta_{d2}\}$ (respectively θ_f) simultaneously. This is summarized in the following minimax problem.

$$\min_{\theta_f, \theta_y} \Big(\mathcal{L}\big(\theta_f, \theta_y; \mathcal{D}(\mathrm{S}, \mathrm{S}')\big) - \max \Big\{ \min_{\theta_{d1}} \mathcal{L}\big(\theta_f, \theta_{d1}; \mathcal{D}(\mathrm{S}, \mathrm{T})\big), \min_{\theta_{d2}} \mathcal{L}\big(\theta_f, \theta_{d2}; \mathcal{D}(\mathrm{S}', \mathrm{T})\big) \Big\} \Big) \tag{1}$$

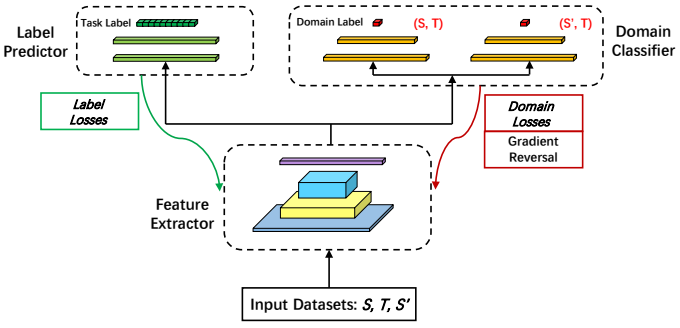

Fig. 1. The model architecture of centralized multi-source UDA [8], consisting of a feature extractor, a label predictor, and two domain classifiers. A gradient reversal layer is applied to the back-propagation through the domain classifiers.

B. Assisted Unsupervised Domain Adaptation

The aforementioned centralized multi-source UDA requires that L has access to all of P's data, which can be unrealistic in practical scenarios. In this subsection, we propose an assisted learning algorithm for unsupervised domain adaptation that enables P to assist L to improve its UDA performance without data sharing via a few interaction rounds.

Specifically, we consider the scenario that L holds labeled source data $\mathcal{D}(\mathrm{S})$ (limited population) and unlabeled target data $\mathcal{D}(\mathrm{T})$. Meanwhile, P holds a large labeled source data $\mathcal{D}(\mathrm{S}')$. We assume the data domains S, T, S' are closely related. To perform assisted UDA without data sharing, L holds a local UDA model (the same as used in [3]) that consists of a feature extractor θ_f, a label predictor θ_y, and a domain classifier θ_d, whereas P holds a local classification model that consists of a feature extractor θ_f and a label predictor θ_y. Notably, P does not need a domain classifier because it does

Algorithm 1 AssistUDA

Input: Initialization model $(\theta_f^0, \theta_y^0, \theta_d^0)$, learning rate η, assistance rounds R, number of local training iterations K (for Learner) and K' (for Provider).

for assistance rounds $r = 1, \ldots, R$ **do**

Learner L:

► Initialize $(\theta_{f_0}^{(\mathrm{L})}, \theta_{y_0}^{(\mathrm{L})}, \theta_{d_0}^{(\mathrm{L})}) = (\theta_f^{r-1}, \theta_y^{r-1}, \theta_d^{r-1})$.

► Train local model by the standard UDA loss $\min_{\theta_f,\theta_y}\big(\mathcal{L}(\theta_f,\theta_y;\mathcal{D}(\mathrm{S})) - \min_{\theta_d}\mathcal{L}(\theta_f,\theta_d;\mathcal{D}(\mathrm{S},\mathrm{T}))\big)$. Obtain a trajectory of UDA models $\{\theta_{f_t}^{(\mathrm{L})}, \theta_{y_t}^{(\mathrm{L})}, \theta_{d_t}^{(\mathrm{L})}\}_{t=1}^{K}$.

► Sample a subset of label prediction models $\{\theta_{f_t}^{(\mathrm{L})}, \theta_{y_t}^{(\mathrm{L})}\}_{t\in\mathcal{T}}$, compute their corresponding classification loss $\{\mathcal{L}(\theta_{f_t}^{(\mathrm{L})}, \theta_{y_t}^{(\mathrm{L})}; \mathcal{D}(\mathrm{S}))\}_{t\in\mathcal{T}}$ and send to P.

Provider P:

► Initialize $(\theta_{f_0}^{(\mathrm{P})}, \theta_{y_0}^{(\mathrm{P})}) = (\theta_{f_{t_*}}^{(\mathrm{L})}, \theta_{y_{t_*}}^{(\mathrm{L})})$ where

$$t_* = \arg\min_{t\in\mathcal{T}} \mathcal{L}(\theta_{f_t}^{(\mathrm{L})}, \theta_{y_t}^{(\mathrm{L})}; \mathcal{D}(\mathrm{S},\mathrm{S}')).$$

► Train local model by $\min_{\theta_f,\theta_y}\mathcal{L}(\theta_f,\theta_y;\mathcal{D}(\mathrm{S}'))$. Obtain a trajectory of label prediction models $\{\theta_{f_t}^{(\mathrm{P})}, \theta_{y_t}^{(\mathrm{P})}\}_{t=1}^{K'}$.

► Sample a subset of label prediction models $\{\theta_{f_t}^{(\mathrm{P})}, \theta_{y_t}^{(\mathrm{P})}\}_{t\in\mathcal{T}'}$, compute their corresponding classification loss $\{\mathcal{L}(\theta_{f_t}^{(\mathrm{P})}, \theta_{y_t}^{(\mathrm{P})}; \mathcal{D}(\mathrm{S}'))\}_{t\in\mathcal{T}'}$. Send these results and t_* to L.

Learner L:

► Output model $(\theta_f^r, \theta_y^r, \theta_d^r)$ where

$$(\theta_f^r, \theta_y^r) = \arg\min_{(\theta_f,\theta_y)\in\{(\theta_{f_t}^{(\mathrm{P})}, \theta_{y_t}^{(\mathrm{P})})\}_{t\in\mathcal{T}'}} \mathcal{L}(\theta_f,\theta_y;\mathcal{D}(\mathrm{S},\mathrm{S}')),$$

and $\theta_d^r = \theta_{d_{t_*}}^{(\mathrm{L})}$.

end

Output: Label prediction model determined by (θ_f^R, θ_y^R).

not have access to the target data $\mathcal{D}(\mathrm{T})$. The overall learning problem is summarized as follows

$$\min_{\theta_f,\theta_y}\Big(\mathcal{L}\big(\theta_f,\theta_y;\mathcal{D}(\mathrm{S},\mathrm{S}')\big) - \min_{\theta_d}\mathcal{L}\big(\theta_f,\theta_d;\mathcal{D}(\mathrm{S},\mathrm{T})\big)\Big), \quad (2)$$

which approximates (1) under the assumption that S and S′ are closely related. We summarize our assisted UDA algorithm (named *AssistUDA*) in Algorithm 1.

To elaborate, the learning process consists of R rounds, each having the following interaction steps between the learner L and the service provider P.

1) First, L starts a local learning process by initializing its feature extractor $\theta_{f_0}^{(\mathrm{L})}$, label predictor $\theta_{y_0}^{(\mathrm{L})}$, and domain classifier $\theta_{d_0}^{(\mathrm{L})}$ from the models obtained from the last round. Then, L applies any standard optimizer (e.g., SGD, Adam) with a proper learning rate η to train local models by minimizing the standard UDA loss for K iterations using the local source and target datasets $\mathcal{D}(\mathrm{S}), \mathcal{D}(\mathrm{T})$. During the training, L stores label prediction models $\{\theta_{f_t}^{(\mathrm{L})}, \theta_{y_t}^{(\mathrm{L})}\}$ sampled from the training trajectory, denoted by $t \in \mathcal{T} \subset \{1, 2, ..., K\}$. Lastly, L sends these models and their corresponding classification losses $\{\mathcal{L}(\theta_{f_t}^{(\mathrm{L})}, \theta_{y_t}^{(\mathrm{L})}; \mathcal{D}(\mathrm{S}))\}_{t\in\mathcal{T}}$ to P.
2) Upon receiving the information from L, P first evaluates the global classification loss $\mathcal{L}(\cdot;\mathcal{D}(\mathrm{S},\mathrm{S}')) = \mathcal{L}(\cdot;\mathcal{D}(\mathrm{S})) + \mathcal{L}(\cdot;\mathcal{D}(\mathrm{S}'))$ of the received set of models $\{(\theta_{f_t}^{(\mathrm{L})}, \theta_{y_t}^{(\mathrm{L})})\}_{t\in\mathcal{T}}$ and picks the one that achieves the minimum global classification loss as the initialization model $(\theta_{f0}^{(\mathrm{P})}, \theta_{y0}^{(\mathrm{P})})$. Notably, $\{\mathcal{L}(\theta_{f_t}^{(\mathrm{L})}, \theta_{y_t}^{(\mathrm{L})}; \mathcal{D}(\mathrm{S}))\}_{t\in\mathcal{T}}$ is already provided by L and $\{\mathcal{L}(\theta_{f_t}^{(\mathrm{L})}, \theta_{y_t}^{(\mathrm{L})}; \mathcal{D}(\mathrm{S}'))\}_{t\in\mathcal{T}}$ can be evaluated from the local data of P. After that, P applies any standard optimizer with a proper learning rate η to train its feature extractor and label predictor by minimizing the label classification loss for K' iterations using the local source dataset $\mathcal{D}(\mathrm{S}')$. During the training, P stores label prediction models $\{\theta_{f_t}^{(\mathrm{P})}, \theta_{y_t}^{(\mathrm{P})}\}$ generated in a subset of the training iterations $t \in \mathcal{T}' \subset \{1, 2, ..., K'\}$. Lastly, P sends these label prediction models and their corresponding classification losses $\{\mathcal{L}(\theta_{f_t}^{(\mathrm{P})}, \theta_{y_t}^{(\mathrm{P})}; \mathcal{D}(\mathrm{S}'))\}_{t\in\mathcal{T}'}$ to L.
3) Once L receives the information sent by P, it evaluates the global classification loss $\mathcal{L}(\cdot;\mathcal{D}(\mathrm{S},\mathrm{S}'))$ of the received set of models $\{(\theta_{f_t}^{(\mathrm{P})}, \theta_{y_t}^{(\mathrm{P})}\}_{t\in\mathcal{T}'}$ and picks the one that achieves the minimum global classification loss as the output model of this assistance round.

Remark. The proposed AssistUDA algorithm does not require data sharing between the learner and the provider. In the local training steps, both the learner and the provider will store a subset of generated label prediction models and evaluate their corresponding classification losses on their respective local source data. Then, these sampled models and their losses are sent to the other side. These information are used to help identify the best model that achieves the minimum global classification loss on the combined source data samples. Our experiments have shown that storing a few models generated in the local training is sufficient to achieve good performance. Also, note that (2) does not contain the loss term $\mathcal{L}(\theta_f, \theta_{d2}; \mathcal{D}(\mathrm{S}', \mathrm{T}))$ in (1), but it is still expected to improve the learner's performance by leveraging the large-sized external data from S', particularly when there is moderate divergence between the domains S and S'.

III. Experiments

A. Experiment Setup

Baselines. In this section, we conduct deep learning experiments to demonstrate the effectiveness of our proposed AssistUDA algorithm. We test the performance of AssistUDA by comparing it with two baseline methods: 1) centralized multi-source UDA, in which L has access to the centralized data $\mathcal{D}(\mathrm{S}, \mathrm{T}, \mathrm{S}')$ (i.e., all the local datasets possessed by L and P) and performs multi-source UDA [8]. Its performance can be regarded as the upper limit; 2) single-source UDA, in

which L performs single-source domain adaption using only its local data $\mathcal{D}(\mathrm{S}, \mathrm{T})$. Its performance can be regarded as the lower limit. In both baseline methods, L adopts the same UDA model as in AssistUDA, which consists of a feature extractor, a label predictor and a domain classifier as described in Subsection II-B.

Datasets. Following [3], [8], we use two sets of datasets, each of which consists of three datasets from closely related domains. Specifically, the first set includes MNIST [22], MNIST-M [7] and SVHN [23], and the second set includes CIFAR-10 [24], CINIC-10 [25] and CINIC-10-grayscaled. With these two sets of datasets, we consider the following four different settings of local datasets of L and P.

TABLE I
LIST OF LOCAL DATASETS OF LEARNER (L) AND PROVIDER (P).

	$\mathcal{D}(\mathrm{S})$	$\mathcal{D}(\mathrm{T})$	$\mathcal{D}(\mathrm{S}')$
Setting 1	SVHN	MNIST	MNIST-M
Setting 2	SVHN	MNIST-M	MNIST
Setting 3	CINIC-10-gray	CIFAR-10	CINIC-10
Setting 4	CINIC-10-gray	CINIC-10	CIFAR-10

Data Distribution. Each of the dataset setting listed in Table I is further specified by a hyperparameter $\gamma \in (0, 1)$ described as follows. First, all the local datasets $\mathcal{D}(\mathrm{S}), \mathcal{D}(\mathrm{T}), \mathcal{D}(\mathrm{S}')$ are balanced, i.e., they contain equal number of samples from each of the 10 classes. Second, both L's target data $\mathcal{D}(\mathrm{T})$ and P's source data $\mathcal{D}(\mathrm{S}')$ are set to have a large sample size, i.e., 46.59k for settings 1 & 2 and 50k for settings 3 & 4. Lastly, the size of L's source data $\mathcal{D}(\mathrm{S})$ is specified by γ times that of $\mathcal{D}(\mathrm{S}')$, and we consider $\gamma = 0.01, 0.1$ so that L has a limited number of source data.

Training hyperparameters. We implement all the algorithms using the standard SGD optimizer with a fixed batch-size 256 and fine-tuned learning rate. Also, as L's source data $\mathcal{D}(\mathrm{S})$ is considerably smaller than its target data $\mathcal{D}(\mathrm{T})$, we duplicate it to match the data size in the UDA training. For centralized multi-source UDA, we define one epoch as one pass over the data. For our AssistUDA, one epoch corresponds to one round of interaction between L and P, where all the local trainings pass the local data once and store the updated local models for every $I = 30$ local training iterations. For single-source UDA, one epoch is defined as two passes over the data so that it uses roughly the same number of training samples as that of the other two algorithms. For all experiments, we perform 10 epochs of training.

B. Experiment Results

Setting 1: SVHN-$\mathcal{D}(\mathbf{S})$, MNIST-$\mathcal{D}(\mathbf{T})$, MNIST-M-$\mathcal{D}(\mathbf{S}')$. We compare all the aforementioned algorithms under setting 1 (see Table I) with $\gamma = 0.01$ (small size $\mathcal{D}(\mathrm{S})$) and $\gamma = 0.1$ (large size $\mathcal{D}(\mathrm{S})$), respectively. Fig. 2 plots the results of training loss (evaluated on $\mathcal{D}(\mathrm{T})$) and test accuracy (evaluated on 10k MNIST test data samples). It can be seen that our AssistUDA consistently achieves a comparable performance to that of the ideal centralized multi-source UDA, and a significantly improved performance over that of single-source UDA. In particular, the performance gain is larger when L has a smaller source dataset (i.e., smaller γ). This shows that our AssistUDA is an effective algorithm that allows the learner to seek assistance from the service provider without data sharing.

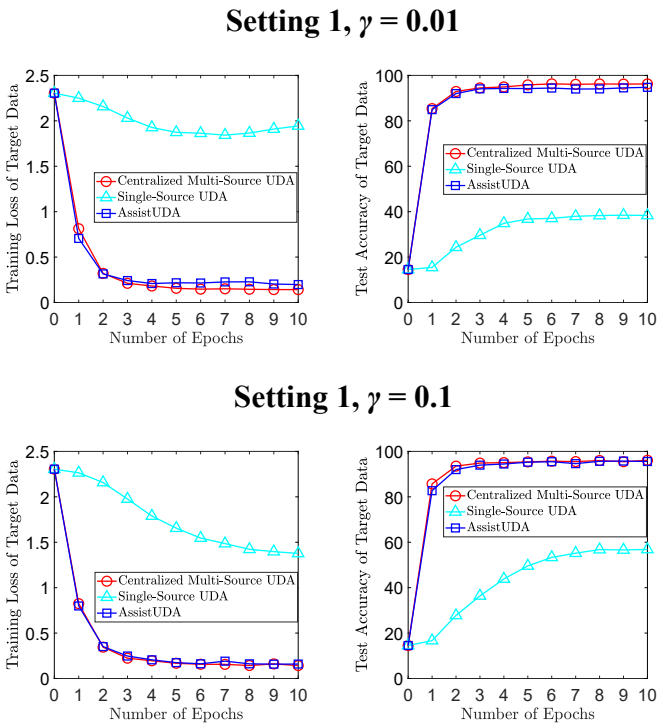

Fig. 2. Comparison of Centralized Multi-Source UDA, Single-Source UDA, and AssistUDA under setting 1 with $\gamma = 0.01$ (top) and 0.1 (bottom).

Setting 2: SVHN-$\mathcal{D}(\mathbf{S})$, MNIST-M-$\mathcal{D}(\mathbf{T})$, MNIST-$\mathcal{D}(\mathbf{S}')$. We further compare these algorithms under setting 2 with $\gamma = 0.01, 0.1$. The results of training loss (evaluated on $\mathcal{D}(\mathrm{T})$) and test accuracy (evaluated on the 9k MNIST-M test data samples) are shown in Fig. 3, where one can make a similar conclusion. Specifically, with $\gamma = 0.01$, the performance of our AssistUDA is slightly worse than that of centralized multi-source UDA, but is much better than that of single-source UDA. When $\gamma = 0.1$ (i.e., $\mathcal{D}(\mathrm{S})$ has a larger size), AssistUDA still achieves a significantly better performance than that of single-source UDA.

Setting 3: CINIC-10-gray-$\mathcal{D}(\mathbf{S})$, CIFAR-10-$\mathcal{D}(\mathbf{T})$, CINIC-10-$\mathcal{D}(\mathbf{S}')$. We repeat the above experiments under setting 3 with $\gamma = 0.01, 0.1$. The results of training loss (evaluated on $\mathcal{D}(\mathrm{T})$) and test accuracy (evaluated on 10k CIFAR-10 test data samples) are shown in Fig. 4. One can make the same observations and conclusions as before from these results, which demonstrate that our AssistUDA works well on diverse types of datasets.

Setting 4: CINIC-10-gray-$\mathcal{D}(\mathbf{S})$, CINIC-10-$\mathcal{D}(\mathbf{T})$, CIFAR-10-$\mathcal{D}(\mathbf{S}')$. Lastly, we test these algorithms under setting 4 with $\gamma = 0.01, 0.1$. Fig. 5 shows the results of

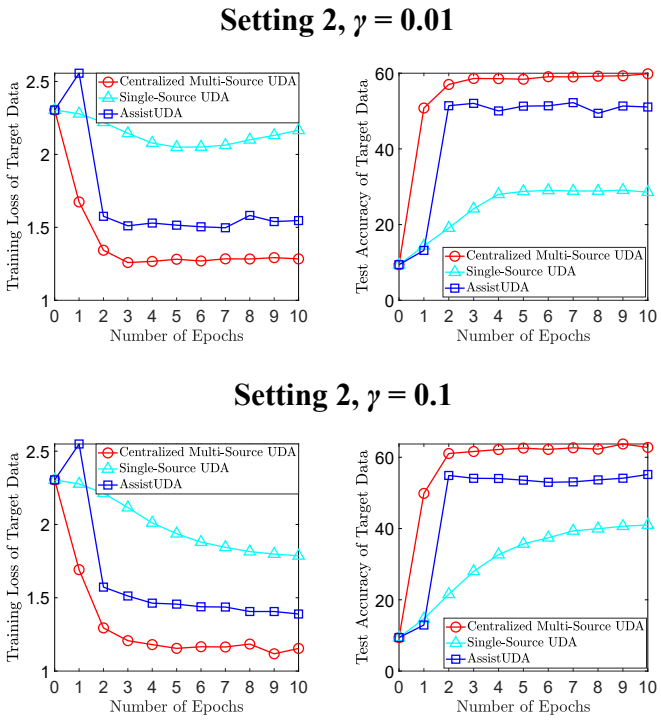

Fig. 3. Comparison of Centralized Multi-Source UDA, Single-Source UDA, and AssistUDA under setting 2 with $\gamma = 0.01$ (top) and 0.1 (bottom).

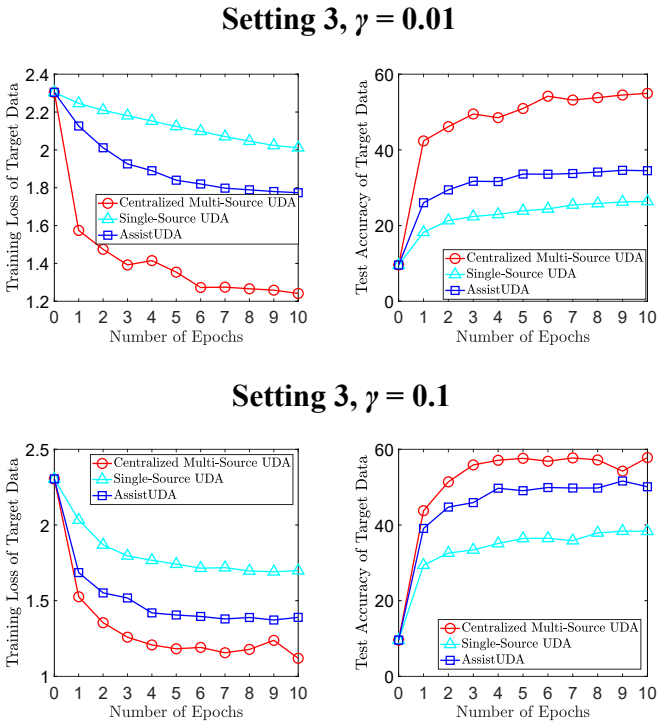

Fig. 4. Comparison of Centralized Multi-Source UDA, Single-Source UDA, and AssistUDA under setting 3 with $\gamma = 0.01$ (top) and 0.1 (bottom).

training loss (evaluated on $\mathcal{D}(\mathrm{T})$) and test accuracy (evaluated on 18k CINIC-10 test data samples). Note that here the CINIC-10 test data samples are generated by sampling uniformly from each class of the original 90k CINIC-10 test samples. These results lead to the same conclusions and prove the effectiveness of AssistUDA on different target domains.

ACKNOWLEDGMENTS

This paper is based upon work supported by the Army Research Laboratory and the Army Research Office under grant number W911NF-20-1-0222 and National Science Foundation under grant number DMS-2134148. The work of Cheng Chen and Yi Zhou was supported in part by U.S. National Science Foundation under the Grant. Nos. CCF-2106216, DMS-2134223 and CAREER-2237830.

IV. CONCLUSION

We propose a fully decentralized assisted learning framework for unsupervised domain adaptation that avoids data sharing between a learner and an external service provider. Our experiments show that the proposed algorithm can assist the learner to achieve a significantly better model domain adaptation performance within a few interaction rounds with the external service provider. We expect that this algorithm can be used to help small companies and individuals who have the need to build high-quality machine learning models but lack a sufficient amount of source data. In the future, an interesting direction is to extend this framework to the multi-learner and multi-provider scenarios.

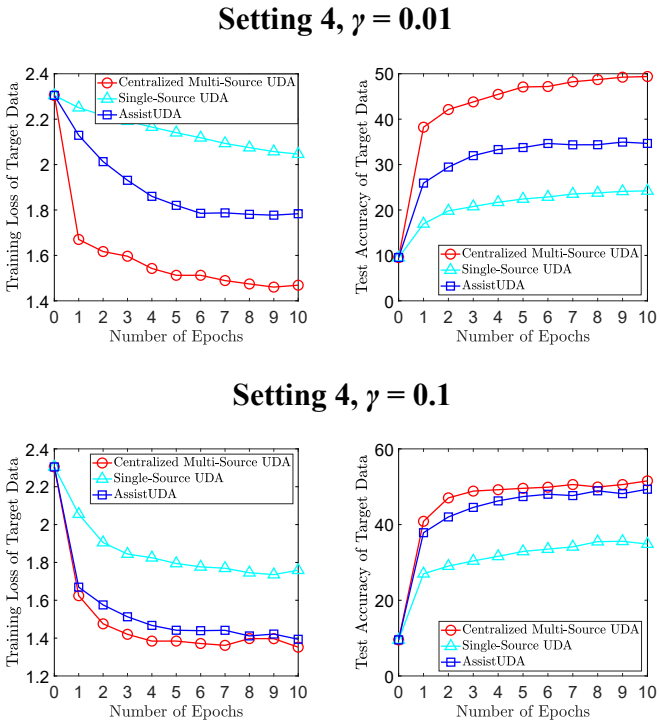

Fig. 5. Comparison of Centralized Multi-Source UDA, Single-Source UDA, and AssistUDA under setting 4 with $\gamma = 0.01$ (top) and 0.1 (bottom).

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, may 2017.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.

[4] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proc. International Conference on Machine Learning*, 2011, p. 513–520.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *Proc. International Conference on Machine Learning*, 2014.

[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc.Neural Information Processing Systems*, 2014, p. 3320–3328.

[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.

[8] H. Zhao, S. Zhang, G. Wu, J. M. F. Moura, J. P. Costeira, and G. J. Gordon, "Adversarial multiple source domain adaptation," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[9] M. Ribeiro, K. Grolinger, and M. A. Capretz, "Mlaas: Machine learning as a service," in *Proc. ICMLA*. IEEE, 2015, pp. 896–902.

[10] X. Xian, X. Wang, J. Ding, and R. Ghanadan, "Assisted learning: A framework for multi-organization learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[11] E. Diao, J. Ding, and V. Tarokh, "GAL: Gradient assisted learning for decentralized multi-organization collaborations," in *Advances in Neural Information Processing Systems*, 2022.

[12] X. Wang, J. Zhang, M. Hong, Y. Yang, and J. Ding, "Parallel assisted learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 5848–5858, 2022.

[13] C. Chen, J. Zhou, J. Ding, and Y. Zhou, "Assisted learning for organizations with limited data," *arXiv preprint arXiv:2109.09307*, 2021.

[14] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.

[16] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," in *International Conference on Learning Representations*, 2021.

[17] ——, "SemiFL: Communication efficient semi-supervised federated learning with unlabeled clients," in *Advances in Neural Information Processing Systems*, 2022.

[18] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[19] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *in Proc. TIST*, vol. 10, no. 2, p. 12, 2019.

[20] Y. Liu, Y. Kang, X. Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient vertical federated learning framework," *arXiv preprint arXiv:1912.11187*, 2019.

[21] T. Chen, X. Jin, Y. Sun, and W. Yin, "Vafl: a method of vertical asynchronous federated learning," *arXiv preprint arXiv:2007.06081*, 2020.

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

[23] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[24] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[25] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "Cinic-10 is not imagenet or cifar-10," *arXiv preprint arXiv:1810.03505*, 2018.