Providing Domain Specific Model via Universal No Data Exchange Domain Adaptation

Lixu Wang\textsuperscript{1,2}, Songtao Liang\textsuperscript{1} and Feng Gao\textsuperscript{*}

\textsuperscript{1} Research Center for Cyber-Physical-Social System, Zhejiang Lab, Hangzhou, Zhejiang, 310000, China

\textsuperscript{2} Computer Science Department, Northwestern University, Evanston, IL, U.S.

* gaof@zhejianglab.com

Abstract. High quality Data and its derivative machine learning (ML) models are gradually becoming commercial commodity and provided to work effectively in an increasing number of areas. These ML model providers possess a set of trained models along with immense amount of source data stored on their servers. In order to obtain specific models, providers will require consumers to upload their domain-specialized data and conduct domain adaptation at the server side. However, considering the protection of the private information reflected by consumers’ training data, as well as maintaining the commercial competitiveness of ML service, it is best that there is no data exchange between servers and consumers. Besides, consumers’ data is always lack of supervision, i.e., classification labels, thus we are searching for how to conduct unsupervised domain adaptation (UDA) with no data exchange among domains in this work. We are the first to propose a novel memory cache based adversarial training (AT) strategy for UDA at the target side without the source data (the existence of source data is an essential requirement for regular AT). And our method includes a multiple pseudo labelling operation which is more accurate and robust than single pseudo labelling. The AT and multiple labelling work collaboratively to extract shared features among domains and adapt the learning model more specific to target domain. We carry out extensive evaluation experiments with a number of data sets and baselines, and according to the results, our proposed method can perform very well and exceed the state-of-art performance on all tasks. In the end, we also discuss how to extend our method to partial and open-set domain adaptation.

1. Introduction
Currently, deep neural networks have shown unparalleled advantages in various domains, attracting increasing interest from both academic and industry areas. However, training a deep neural network and then applying it requires expensive resources: laborious manual annotated data, powerful GPU and enough memory space. Some giant tech companies, like Google and Amazon, release a number of well-trained machine learning (ML) models to promote the development and application of Artificial Intelligence. In the future, we have faith that neural networks with great performance will be regarded as commercial commodity and provided by these companies as a kind of service. For this service, consumers are only required to provide the training data from their specific domains. Practically, such data is usually unlabelled due to heavy workload of manual annotation and also such annotation operations require for expert knowledge. In order to address the unsupervised problem, ML service providers require consumers to upload their training data, then label it and train ML models at the server side. However, the training data at the network edge is continuously generated, thus real-time data
exchange between the edge and server will induce heavy communication overhead, and much worse, real-time annotation for the newly generated data is inaccurate and unreliable due to the discrepancy among data from different domains. In addition, considering the private information embedded in people’s training data, uploading the private data exposes a vulnerable channel for adversarial attackers to eavesdrop and steal consumers’ privacy. In order to address the issue of communication workload reduction and privacy leakage, remaining the training data at the consumer side and conducting local model training with the instruction from the server are promising solutions, e.g., federated learning (FL) is the most representative strategy. However, the standard FL is inapplicable in the aforementioned case of unsupervised learning. Intuitively, the server can send a teacher model to the edge and apply it to annotate such unlabelled data, then the client can use the data to conduct fine-tuning or distillation from the teacher model and get the student model. But in practice, the feature space of the edge data possibly has considerable discrepancy with that of training data learned by the teacher model. In this case, the annotation from the teacher model is inaccurate, degrading the performance of student model. Fortunately, Unsupervised Domain Adaptation (UDA) is proposed to shrink domain discrepancy and search for a universally applicable model among domains. Conducting standard UDA needs the co-existence of source and target data, but considering the user privacy and communication efficiency, the service provider cannot require clients to upload their raw data [1]–[3], meanwhile, clients don’t have permission to access the data stored on the service server. Therefore, it’s necessary to propose new approaches to adapt knowledge from source domain to target domain in the unsupervised scenario of no data exchange.

There are several works about conducting unsupervised adaptation in the target domain without the access of the source data [2], [3]. However, these works have following drawbacks: 1) to replace the source data, there are a number of additional modules which can generate source-like samples in the target model training, which will induce much more computation burden on the user side; 2) the performance requirement of source models is increasingly much higher than regular situations, i.e., source models will be re-trained by a large amount of additional synthesized data to perform well in a much wider feature distribution, in this way, the transfer ability of source models is strengthened to work effectively in the case of no source data. These issues will induce serious limitations in the case of ML service, thus we need to propose simpler UDA method which has weaker requirements for both the source and target domains and meanwhile remains effective in the setting of no data exchange.

In this paper, we propose a simple but effective source-free domain adaptation approach. For our method, all adaptation process is carried out at the target side, and we won’t incorporate many additional modules like previous arts but only a domain discriminator, in this case, the computation workload at the target is relatively small. Further, the training of source models is similar to regular supervised learning, and we do not synthesize or generate additional source data to augment the performance of source models. In order to defeat the domain discrepancy, we propose an original adversarial training strategy in the setting of no source data, and for learning the target knowledge, we design a multiple pseudo labelling method to annotate the unlabelled target data. We carry out extensive experiments on two popular domain adaptation data sets, and according to the results, our method can perform very well in the setting of no source data and even exceed the state-of-art performance with the access of source data.

Specifically, our contribution can be summarized as follows:

- We are the first to propose an adversarial training-based domain adaptation method without the access of source data.
- We propose a multiple pseudo labelling approach, which can tolerate mistakes of single pseudo labelling. This approach will make the pseudo labelling results more robust and accurate.
- We conduct extensive experiments on popular domain adaptation data sets and compare our proposed method with a number of baseline approaches. The results show that our method can achieve the state-of-art performance.
2. Related Work

2.1. Unsupervised Domain Adaptation
UDA is trying to adapt pre-trained models trained with labelled source data to a new model which will be applied in specific domains with unlabelled data. Previously arts design a number of losses to quantify the discrepancy between source and target data distributions [4]–[7]. Afterwards, the principle of adversarial battle in Generator Adversarial Network (GAN) [8] is proposed by the deep learning community, and UDA is also benefited from it. Different from the adversarial training in GAN, the battle parties in UDA are feature extractor and domain discriminator. After the battle between these two parties, the extractor can extract the shared high-level features of both the source and target domains, and the state-of-art performance of UDA has been improved to a much higher level. Considering different overlapping settings of source and target domains, UDA can be classified into Closed-set [9], [10], Partial [11], [12] and Open-set [1], [13] domain adaptations. Closed-set DA is based on that the label space of the target domain is equal to that of the source domain, thus every source label corresponds to a target label. For Partial DA, the label space of target domain is smaller than that of source domain, and the primary concern is suppressing the negative transfer from the source labels which are not shared with target domain. Open-set DA is built on more practical scenarios, i.e., there are target samples which do not belong to any source label and thus called as unknown labels, the target model is required to classify unknown and known labels meanwhile. The toughest mission of Open-set DA is restricting the distribution boundary between unknown and known labels. Providing that a method that is applicable for all of these 3 adaptation tasks, it will be regarded as a Universal UDA solution [14], [15]. Our work mainly focuses on vanilla Closed-set DA, but we also discuss how to adjust the proposed method to Partial and Open-set DA.

2.2. Privacy Protection
Recently, more and more adversarial attacks [16]–[18] against machine learning (ML) are proposed, raising people’s awareness of protecting their private information when utilizing artificial intelligence applications. For instance, people will be unwilling to share their private data to giant tech companies, thus there are reasons to believe the training of ML models needs to weaken the requirement of accessing people’s private data in the future. Considering this, UDA problem also should not require data exchange between source and target domains. There are several exploring works: [1] designs a framework to identify samples that exclude from both the source and target domain without data exchange between domains. [2] proposes an approach that enables target adaptive models to be deployed well only with the access of source model. These two works synthesize more source data by feature exchange and combination to augment the source training, strengthening the transfer ability of source model. Another work utilizes generative models to generate source-like data at the target domain, then accesses both the generated and target data to conduct domain adaptation. The simplest data-free arts [3] without data augmentation are based on the hypothesis feature transfer from batch-normalized statistics, though there is a little performance reduction compared with the augmentation-based method.

Federated learning is a promising approach to train ML models without the access of raw data, and there are also explorations to make domain adaptation federated [19], [20]. But these works are searching for building a better global target model by aggregating the knowledge of multiple source domains, which has difference with our problem of interest.

3. Methodology

3.1. Numbering
Our work mainly considers the task of image classification, let us define the distribution of the labelled data \((D_s) = (x_s^i, y_i)\) which is stored on the server of Machine Learning service providers as the source domain, in addition, there is a source model \(M_s\) trained with \(D_s\) which has perfect performance in the source domain. While at the user side, at the start, there is only the unlabelled target data \(D_t = \{x_t^i\}\).
The final objective of UDA is adapting $M_s$ to a new target model $M_t$ which performs well in the target domain, and under the specific case of aforementioned scenarios, there is not any data exchange between the source and target domain during adaptation.

### 3.2. Memory-Based Adversarial Training

Following the basic workflow, the client will receive a model as the initial target model $M_{t(i)}$ to conduct local domain adaptation with unlabelled target data. $M_{t(i)}$ is a copy of $M_s$, thus it stores the knowledge of source domain, next we will introduce how we extract the overlapping knowledge between the source and the target domain. In order to adapt feature encoder $E$ and classifier $C$ to the target domain, the primary step is making them easier to distinguish domain-shared features and domain-specific ones, and we achieve this by a novel adversarial training strategy. We know that the final output of $E$ is the high-level features of training data, and if we add a domain discriminator $D_E$ to classify these high-level features into target and source domain, then we can force $E$ to extract the shared high-level features by conducting adversarial battle between $E$ and $D_E$. However, unlike the regular adversarial training in UDA, our method is applicable in the case of no access for the source data. To be specific, in the source training of $M_s$, we label the source domain as the positive, and use $(D_s)$ to train $E \& D_E$ with the following adversarial loss:

$$L^s_{DD} = \frac{1}{n_s} \sum_{i=1}^{n_s} L_{CE} \{D_E(E(x^s_i)), 1\} \tag{3.1}$$

where $n_s$ is the sample number of source data, $L_{CE}$ represents the Batch CrossEntropy loss. In addition to $L^s_{DD}$, the training of the classifier $C$ at the source side still needs to optimize with classification loss of $(D_s)$, then the overall loss $L^s$ in the source training is written as:

$$L^s = -\frac{1}{n_s} \sum_{i=1}^{n_s} \{y_i^{(s)} \cdot \log(p(y_i|x^s_i))\} + L^s_{DD} \tag{3.2}$$

here we conduct a label smoothing operation [21] in the source training, i.e., $y_i^{(s)} = (1-\alpha)\bar{y} + \alpha/Q$ where $Q$ is the number of classes and $\alpha$ is the smoothing control parameter which is empirically set to 0.1. The label smoothing will make the source model output more reliable classification results with low uncertainty, which will help the adaptation in the target domain.

After the training, $D_E$ can discriminate the source and non-source domains, but cannot distinguish target and source domains. Next, $D_E$, along with $M_{t(i)}$ will be sent to clients from the server, then clients use $(D_s)$ to conduct adaptation at the target side. The adaptation process is regularized by a loss which consists of four parts: domain discrimination (DD), information entropy (IE), pseudo labelling (PL) and sub-labelling (SL). For DD, we regard $(D_s)$ as the negative domain, then the loss part for the original target data is formulated as:

$$L^{t(o)}_{DD} = \frac{1}{n_t} \sum_{i=1}^{n_t} L_{CE} \{D_E(E(x^t_i)), 0\} \tag{3.3}$$

here, $n_t$ is the sample number of $(D_s)$. In the source training, discriminator $D_E$ doesn’t battle with extractor $E$, but at the target side, we will place a gradient reverse layer (GRL) [22] between $D_E$ and $E$, then these two modules will battle with each other gradually. The objective of $E$ is making its extracted high-level features to confuse the discriminability of $D_E$ in terms of domain belongings, while the task of $D_E$ is distinguishing the inputting features from the source to the target domain accurately. The GRL achieves this objective by incorporating a negative coefficient $\lambda$ to its received loss gradient during back-propagation, and $\lambda$ is written as:
\[
\lambda = 1 - \frac{2}{1 + e^{-10\gamma}}
\] (3.4)

Here, \( \gamma \) is an index indicating the progress of adversarial training, equals to the ratio between the current round and the overall number of rounds. With the adversarial battle, extractor \( E \) will tend to extract the shared features between source and target domains in the end.

However, the \( D_E \)’s memory for the source features is fading away during the adversarial battle, thus we need to find ways to make \( D_E \) remain its distinguishing ability for the source domain. We design a cache to store the high-level features which are similar to the source domain. As stated above, the task of \( D_E \) is binary classification, and all target samples are regarded as the negative. We can push the target high-level features that make \( D_E \) incorrectly classify to the cache at every round. The adversarial battle is controlled by a global round index \( \gamma \), the higher the \( \gamma \) is, the stiffer the battle between \( E \) and \( D_E \) will be. In other words, the domain discrimination at the target side can be divided into two phases: when \( \gamma \) is low, the training enables \( D_E \) to distinguish target and non-target domains; when \( \gamma \) is high, \( D_E \) can distinguish source and target domains if its memory for the source domain remains, then with the drive of \( \gamma \), the current training will make \( D_E \) confuse to distinguish between the source and target domains. At every target training round, there might always be misclassified features, but due to the source memory reduction, we would like to pay more attention to such misclassified features at rounds of low \( \gamma \). As a result, we will use this memory cache progressively as follows:

\[
L_{DD}^{(m)} = \frac{1}{n_m} \sum_{i=1}^{n_m} (1 - \gamma) \cdot L_{CE} \{D_E(f_i), 1\}
\] (3.5)

where \( n_m \) is the overall number of feature map samples stored in the cache, while \( f_i \) denotes every cache sample, and with the multiplication with \( (1 - \gamma) \), the samples added in the cache early will contribute more to \( L_{DD}^{(m)} \). Finally, the overall loss for domain discrimination is the sum of aforementioned two parts:

\[
L_{DD} = L_{DD}^{(0)} + L_{DD}^{(m)}.
\]

3.3. Multiple Pseudo Labelling Adaptation

At the beginning of adaptation, \( C \) trained with \( \{D_i\} \) performs badly on extracted high-level features of \( \{D_i\} \), and the main reason is the feature confusion caused by the entanglement of domain-shared and target-specific features. Information entropy is a metric that measure the turbulence of data, and we can utilize it to quantify the uncertainty of classifier prediction. Then the IE loss can be written as:

\[
L_{IE} = \frac{1}{n_t} \sum_{i=1}^{n_t} H(x_t^i) = -\frac{1}{n_t} \sum_{i=1}^{n_t} \sum_{j=1}^{Q} p(y^j|x_t^i) \log(p(y^j|x_t^i))
\] (3.6)

\( H(\cdot) \) denotes computing the information entropy, and \( Q \) is the number of classes.

Subsequently, we will introduce our pseudo labelling algorithm for unlabelled target data. Similar to weighted K-Means, we design a clustering approach based on aforementioned high-level features. First, we need to find the centroid of each class by summing up all high-level features which are weighted by the probability results sample-wisely, finally these summed features will be normalized by dividing the summed probability results. In addition, due to the bad performance of \( C \) at the start of training, it’s reasonable to make the target model to focus more on samples which are more convincing, thus we use entropy to measure the reliability of input samples and weigh them before obtaining the centroids. And this pseudo labelling process can be written as:

\[
C = \frac{\sum_{i=1}^{n_t} \{H(x_t^i) \cdot C(E(x_t^i)) \cdot E(x_t^i)\}}{\sum_{i=1}^{n_t} \{C(E(x_t^i))\} \cdot \sum_{i=1}^{n_t} \{H(x_t^i)\}}
\] (3.7)
And the computation of above equation will get a matrix $C$ with the size of $Q \times F$ where $F$ is the number of high-level feature dimensions. The centroid of class $i$ corresponds to the $i$th row of $C$. After obtaining the centroids, we need to compare each sample with all these centroids and find the closest one to be the pseudo label. Unlike the previous work which applies cosine similarity to compare samples and centroids, we use Mahalanobis Distance (MD) \cite{23} to measure the similarity. Compare to cosine similarity (CS), MD will analyse the relation among different high-level feature components statistically and then compare their similarity, while CS treats different high-level feature components equally thus it cannot find their relations. Meanwhile, MD is also applicable to unnormalized data as the same like CS. Specifically, the computation of MD is written as:

$$
MD(x^t_i, C_i) = \sqrt{(\mathcal{E}(x^t_i) - C_i)^T \cdot \Sigma^{-1} \{\mathcal{E}(x^t_i) - C_i\}}
$$

(3.8)

here $\Sigma$ is the covariance matrix of $\{\mathcal{E}(x^t_i) - C_i\}$ column-wisely. After computing the MD of a sample $x^t_i$ with all centroids, we regard the centroid that is closest to $x^t_i$ as its pseudo label $\hat{y}_i$. With pseudo labels, we can construct the PL Loss as:

$$
L_{PL} = -\frac{1}{n_t} \sum_{i=1}^{n_t} \{\hat{y}_i \cdot \log(p(\hat{y}_i|x^t_i))\}
$$

(3.9)

When computing the MD of sample $x^t_i$, we will get $Q$ MD distances, each MD corresponds to a centroid. Other than just focusing on the minimum distance $d_{m1}$, our work also pays attention to the second smallest one $d_{m2}$, in this case, we design a loss part about sub-labelling. Similar to the part of PL, SL is in the form of CrossEntropy Loss, and we weigh different samples with their second smallest MD distance:

$$
L_{SL} = -\frac{1}{n_t} \sum_{i=1}^{n_t} d_{m1}^{\frac{1}{d_{m2}}} \cdot \hat{y}_i \cdot \log(p(\hat{y}_i|x^t_i))
$$

(3.10)

here $\hat{y}_i$ denotes the label whose centroid is the second closest to $x^t_i$. The addition of SL Loss is beneficial for the robustness of PL, and also can relieve the dependency of correct pseudo labelling. Considering the uncertainty of samples, $d_{m1}/d_{m2}$ will make samples with low uncertainty contribute slightly to the SL Loss and believe the pseudo label more firmly. Above all, after calculating all loss parts, we can compose the whole loss in the target training as:

$$
L^t = L_{ID}^t + L_{IE} + L_{PL} + L_{SL}
$$

(3.11)

And the adaptation process at the target side only needs to input the unlabelled target data and stored high-level features in the memory cache.

4. Evaluation

4.1. Experiment Settings

We use two data sets to evaluate our proposed framework: Office31 \cite{24} and Office-Home \cite{25}. Office31 is a standard benchmark data set which contains three domains: Amazon (A), DSLR (D), Webcam (W), and each domain includes 31 classes of objects. Office-Home is much larger and more challenging than Office31. Office-Home consists of 4 distinct domains, i.e., Artistic images (Ar), Clip Art (Cl), Product images (Pr), and Real-World images (Rw). For each domain, there are 65 categories to be classified. For both Office31 and Office-Home, the basic neural architectures are the same. We apply ResNet-50 as the backbone of feature encoder, and the only difference is we cut off the layers after the last res-block. The main part of classifier consists of a Max-Pool layer, a Batch-Normalization layer and two fully-connected linear layers. As for domain discriminator DE, we use 3 sequential linear blocks (a fully-connected layer, a ReLU activation, and a dropout layer) and a Sigmoid activation layer.
We implement the main code in PyTorch, and use some machine learning functions of Sklearn. All experiments are conducted on a server running Ubuntu 18.04 LTS, equipped with a 2.10GHz CPU Intel Xeon (R) Gold 6130, 64GB RAM, and NVIDIA TITAN RTX GPU cards.

4.2. Main Results
For Office31, we conduct adaptation in all cases of domain pairs: A-D, A-W, D-A, D-W, W-A, W-D, and the experiment results are shown in Table 1. And for results of Office-Home shown in Table 12, there are 12 distinct domain pairs and we also carry out experiments in all cases. In order to illustrate the effectiveness of our method, we compare our performance with a number of state-of-art baselines, including both source-free and source-access approaches, and they are ResNet-50 [26], DANN [22], DAN [27], CDAN [28], rRevGrad+CAT [29], DSBN+MSTN [30], SAFN+ENT [31], CDAN+BNM [32], MDD [33], CDAN+TransNorm [10], CDAN+BNM [34], GVB-GD [35] (Source-Access), and SHOT [3], SHOT++ [36], Batch-Norm [ICLR 2021 submission] (Non-Source).

According to Tables 1 and 2, the performance of Adv-M can exceed averagely all baseline methods. Specifically, Adv-M outperforms the state-of-art for source-access method (GVB-GD) on Office31 by 0.1%, and also for the art for non-source method (SHOT++). And for Office-Home, our method improves the best performance of source-access methods by 2.6% averagely.

Table 1. The performance of different methods on Office31 for closed-set domain adaptation

| Source | Method          | A → D  | A → W  | D → A  | D → W  | W → A  | W → D  | Avg.  |
|--------|----------------|--------|--------|--------|--------|--------|--------|-------|
| Access | ResNet-50      | 68.9   | 68.4   | 62.5   | 96.7   | 60.7   | 99.3   | 76.1  |
|        | DAN            | 78.6   | 80.5   | 63.6   | 97.1   | 62.8   | 99.6   | 80.4  |
|        | DANN           | 79.7   | 82.0   | 68.2   | 96.9   | 67.4   | 99.1   | 82.2  |
|        | SAFN+ENT       | 90.7   | 90.1   | 73.0   | 98.6   | 70.2   | 99.8   | 87.1  |
|        | rRevGrad+CAT   | 90.8   | 94.4   | 72.2   | 98.0   | 70.2   | 100.0  | 87.6  |
|        | CDAN           | 92.9   | 94.1   | 71.0   | 98.6   | 69.3   | 100.0  | 87.7  |
|        | DSBN+MSTN      | 92.2   | 92.7   | 71.7   | 99.0   | 74.4   | 100.0  | 88.3  |
|        | CDAN+BSP       | 93.0   | 93.3   | 73.6   | 98.2   | 72.6   | 100.0  | 88.5  |
|        | CDAN+BNM       | 92.9   | 92.8   | 73.5   | 98.8   | 73.8   | 100.0  | 88.6  |
|        | MDD            | 93.5   | 94.5   | 74.6   | 98.4   | 72.2   | 100.0  | 88.9  |
|        | CDAN+TransNorm | 94.0   | 95.7   | 73.4   | 98.7   | 74.2   | 100.0  | 89.3  |
|        | GVB-GD         | 95.0   | 94.8   | 73.4   | 98.7   | 74.7   | 100.0  | 89.7  |

Table 2. The performance of different methods on Office-Home for closed-set domain adaptation.

| Source | Method          | A' → C' | A' → P' | A' → R | A' → C' | A' → P' | A' → R | C' → A | C' → P | C' → R | Avg.  |
|--------|----------------|---------|---------|--------|---------|---------|--------|--------|--------|--------|-------|
| Access | ResNet-50      | 40.0    | 45.3    | 46.5   | 35.6    | 34.5    | 37.1   | 40.7   | 36.3   | 37.6   | 47.9  |
|        | DAN            | 50.2    | 50.4    | 50.2   | 50.2    | 50.2    | 50.2   | 50.2   | 50.2   | 50.2   | 50.2  |
|        | DANN           | 47.6    | 46.5    | 46.5   | 46.5    | 46.5    | 46.5   | 46.5   | 46.5   | 46.5   | 46.5  |
|        | SAFN+ENT       | 50.2    | 50.2    | 50.2   | 50.2    | 50.2    | 50.2   | 50.2   | 50.2   | 50.2   | 50.2  |
|        | CDAN+BSP       | 50.2    | 50.2    | 50.2   | 50.2    | 50.2    | 50.2   | 50.2   | 50.2   | 50.2   | 50.2  |
|        | CDAN+BNM       | 50.2    | 50.2    | 50.2   | 50.2    | 50.2    | 50.2   | 50.2   | 50.2   | 50.2   | 50.2  |
|        | MDD            | 50.2    | 50.2    | 50.2   | 50.2    | 50.2    | 50.2   | 50.2   | 50.2   | 50.2   | 50.2  |
|        | CDAN+TransNorm | 50.2    | 50.2    | 50.2   | 50.2    | 50.2    | 50.2   | 50.2   | 50.2   | 50.2   | 50.2  |
|        | GVB-GD         | 50.2    | 50.2    | 50.2   | 50.2    | 50.2    | 50.2   | 50.2   | 50.2   | 50.2   | 50.2  |

5. Discussion
5.1. Partial Domain Adaptation
We also discuss how to extend our Adv-M method to partial-set domain adaptation (PDA). For PDA problem, the label space in target domain is smaller than that of source domain, in our method, we only need to discard tiny centroids which should be regarded as empty. Setting a threshold $\beta_{th}$, removing the
centroids with size smaller than $\epsilon_{fate}$ and finally conducting adaptation on the remaining centroids can solve the PDA problem effectively.

5.2. Open-set Domain Adaptation
For the problem of open-set domain adaptation (OSDA), there are target samples that don’t belong to any source label and are called unknown samples. These unknown samples are located outside the known target feature space, and they are possibly adjacent to any target label. During adaptation, the prediction of known samples will be more and more reliable while that of unknown samples still remain uncertain. In order to distinguish known and unknown samples, we apply a binary K-Means clustering to the entropy score of all target samples at each target training round. And we regard samples in the cluster with higher entropy as the unknown, then these unknown samples will be excluded from obtaining the class centroids in the next round.

6. Conclusion
We present a simple, novel and effective method to shrink the performance gap of machine learning (ML) models in various domains, while considering the privacy protection and communication efficiency in future ML service. Our approach is the first to extend the adversarial training strategy to cases of no data exchange among domains via a craft designed domain memory cache. And we also design an original multiple pseudo labelling operation, which is more accurate and robust than previous arbitrary single pseudo labelling. Extensive experiments demonstrate that our method Adv-M can significantly outperform previous related works in its average classification accuracy on Office-31 and Office-Home in the cases of no data exchange. Interestingly, Adv-M can even exceed the state-of-art performance of the methods with the access to the source data.

Acknowledgments
We appreciate the supporting help and time reading of all anonymous reviewers. And we gratefully acknowledge the support from Zhejiang Lab and Zhejiang Province.

References
[1] J. N. Kundu, N. Venkat, A. Revanur, R. V. Babu et al., “Towards inheritable models for open-set domain adaptation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12 376–12 385.
[2] J. N. Kundu, N. Venkat, R. V. Babu et al., “Universal source-free domain adaptation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4544–4553.
[3] J. Liang, D. Hu, and J. Feng, “Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation,” arXiv preprint arXiv:2002.08546, 2020.
[4] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, “Maximum classifier discrepancy for unsupervised domain adaptation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3723–3732.
[5] M. Long, H. Zhu, J. Wang, and M. I. Jordan, “Unsupervised domain adaptation with residual transfer networks,” in Advances in neural information processing systems, 2016, pp. 136–144.
[6] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” arXiv preprint arXiv:1412.3474, 2014.
[7] W. Zellinger, T. Grubinger, E. Lughofers, T. Natschläger, and S. Saminger-Platz, “Central moment discrepancy (cmd) for domain invariant representation learning,” arXiv preprint arXiv:1702.08811, 2017.
[8] M.-Y. Liu and O. Tuzel, “Coupled generative adversarial networks,” in Advances in neural information processing systems, 2016, pp. 469–477.
[9] S. Srinivas and F. Fleuret, “Knowledge transfer with jacobian matching,” arXiv preprint arXiv:1803.00443, 2018.
[10] X. Wang, Y. Jin, M. Long, J. Wang, and M. I. Jordan, “Transferable normalization: Towards improving transferability of deep neural networks,” in Advances in Neural Information Processing Systems, 2019, pp. 1953–1963.

[11] Z. Chen, C. Chen, Z. Cheng, B. Jiang, K. Fang, and X. Jin, “Selective transfer with reinforced transfer network for partial domain adaptation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12 706–12 714.

[12] J. Liang, Y. Wang, D. Hu, R. He, and J. Feng, “A balanced and uncertainty-aware approach for partial domain adaptation,” arXiv preprint arXiv:2003.02541, 2020.

[13] K. Saito, S. Yamamoto, Y. Ushiku, and T. Harada, “Open set domain adaptation by backpropagation,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 153–168.

[14] H. Tang and K. Jia, “Discriminative adversarial domain adaptation.” In AAAI, 2020, pp. 5940–5947.

[15] Y. Jin, X. Wang, M. Long, and J. Wang, “Minimum class confusion for versatile domain adaptation,” in European Conference on Computer Vision. Springer, 2020, pp. 464–480.

[16] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, “Targeted backdoor attacks on deep learning systems using data poisoning,” arXiv preprint arXiv:1712.05526, 2017.

[17] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017, pp. 3–18.

[18] L. Wang, S. Xu, X. Wang, and Q. Zhu, “Eavesdrop the composition proportion of training labels in federated learning,” arXiv preprint arXiv:1910.06044, 2019.

[19] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, “Federated adversarial domain adaptation,” arXiv preprint arXiv:1911.02054, 2019.

[20] D. Peterson, P. Kanani, and V. J. Marathe, “Private federated learning with domain adaptation,” arXiv preprint arXiv:1912.06733, 2019.

[21] R. M. Müller, S. Kornblith, and G. E. Hinton, “When does label smoothing help?” in Advances in Neural Information Processing Systems, 2019, pp. 4694–4703.

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

[23] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, “The mahalanobis distance,” Chemometrics and intelligent laboratory systems, vol. 50, no. 1, pp. 1–18, 2000.

[24] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains,” in European conference on computer vision. Springer, 2010, pp. 213–226.

[25] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, “Deep hashing network for unsupervised domain adaptation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5018–5027.

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[27] M. Long, Y. Cao, J. Wang, and M. Jordan, “Learning transferable features with deep adaptation networks,” in International conference on machine learning. PMLR, 2015, pp. 97–105.

[28] M. Long, Z. Cao, J. Wang, and M. I. Jordan, “Conditional adversarial domain adaptation,” in Advances in Neural Information Processing Systems, 2018, pp. 1640–1650.

[29] Z. Deng, Y. Luo, and J. Zhu, “Cluster alignment with a teacher for unsupervised domain adaptation,” in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 9944–9953.

[30] W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han, “Domainspecific batch normalization for unsupervised domain adaptation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7354–7362.
[31] R. Xu, G. Li, J. Yang, and L. Lin, “Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation,” in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1426–1435.

[32] X. Chen, S. Wang, M. Long, and J. Wang, “Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation,” in International Conference on Machine Learning, 2019, pp. 1081–1090.

[33] Y. Zhang, T. Liu, M. Long, and M. I. Jordan, “Bridging theory and algorithm for domain adaptation,” arXiv preprint arXiv:1904.05801, 2019.

[34] S. Cui, S. Wang, J. Zhuo, L. Li, Q. Huang, and Q. Tian, “Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3941–3950.

[35] S. Cui, S. Wang, J. Zhuo, C. Su, Q. Huang, and Q. Tian, “Gradually vanishing bridge for adversarial domain adaptation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12 455–12 464.

[36] J. Liang, D. Hu, Y. Wang, R. He, and J. Feng, “Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer,” arXiv preprint arXiv:2012.07297, 2020.