Deep Domain Adaptation With Differential Privacy

Qian Wang, *Senior Member, IEEE*, Zixi Li, Qin Zou, *Senior Member, IEEE*, Lingchen Zhao, and Song Wang, *Senior Member, IEEE*

***Abstract*— Nowadays, it usually requires a massive amount of labeled data to train a deep neural network. When no labeled data is available in some application scenarios, domain adaption can be employed to transfer a learner from one or more source domains with labeled data to a target domain with unlabeled data. However, due to the exposure of the trained model to the target domain, the user privacy may potentially be compromised. Nevertheless, the private information may be encoded into the representations in different stages of the deep neural networks, i.e., hierarchical convolutional feature maps, which poses a great challenge for a full-fledged privacy protection. In this paper, we propose a novel differentially private domain adaptation framework called DPDA to achieve domain adaptation with privacy assurance. Specifically, we perform domain adaptation in an adversarial-learning manner and embed the differentially private design into specific layers and learning processes. Although applying differential privacy techniques directly will undermine the performance of deep neural networks, DPDA can increase the classification accuracy for the unlabeled target data compared to the prior arts. We conduct extensive experiments on standard benchmark datasets, and the results show that our proposed DPDA can indeed achieve high accuracy in many domain adaptation tasks with only a modest privacy loss.**

***Index Terms*— Domain adaptation, privacy preservation, differential privacy, deep learning, convolutional neural network.**

I. Introduction

DEEP neural networks have demonstrated an unprecedented strong power in solving many problems in computer vision, natural language processing, and speech recognition, etc. To achieve better generalization and higher performance, large-scale datasets are commonly required during the training. However, it is often labor-intensive and time-consuming to manually label massive data from diverse application domains. In addition, dataset biases usually make it difficult to apply a trained model to unseen data with a different distribution. This motivates the development of unsupervised domain adaptation, a novel technique that tries to transfer the model trained on a dataset from the *source domain* to another unlabeled dataset from the *target domain*.

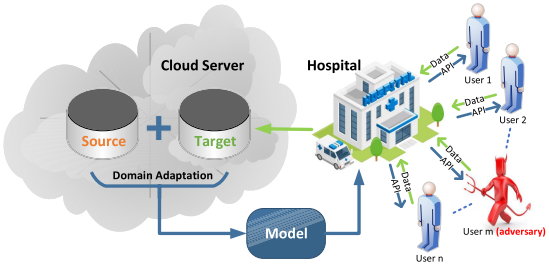


Fig. 1. An illustration of domain adaptation and the risk of privacy leakages.

To be specific, domain adaptation aims to map both the source-domain data and target-domain data into a common feature space and make the classifier trained from the source domain capable of classifying the unlabeled target-domain data. However, there exists a serious privacy concern for the training data in many application scenarios such as hospitals and schools (the owners of the target data), since the sensitive private data used as input for training domain adaptation models can be leaked to others. Some attacks have been proposed to infer sensitive information about the training data from the released trained model [1]–[4].

Considering a practical scenario as illustrated in Figure 1, a hospital receives unlabeled medical data of different users and expects to make a classification or prediction on these data. Since the hospital only has few labeled data, it cannot train a model with high performance based on its own data. Therefore, it sends the unlabeled medical data to a cloud server which has a large amount of labeled medical data. Then, the cloud server trains the model with the unlabeled target-domain data from the hospital and its own source-domain data together by a domain adaptation method. Finally, the trained model is sent back to the hospital and accessible to users. If there is a malicious user, he/she may try to extract private information about the training data from the model. Many attacks only require the knowledge of the

Manuscript received June 25, 2019; revised December 23, 2019; accepted March 5, 2020. Date of publication March 25, 2020; date of current version April 15, 2020. The work of Qian Wang was supported in part by the NSFC under Grant 61822207 and Grant U1636219, in part by the Equipment Pre-Research Joint Fund of Ministry of Education of China (Youth Talent) under Grant 6141A02033327, in part by the Outstanding Youth Foundation of Hubei Province under Grant 2017CFA047, and in part by the Fundamental Research Funds for the Central Universities under Grant 2042019kf0210. The work of Qin Zou was supported in part by the NSFC under Grant 61872277 and in part by the Natural Science Foundation of Hubei Province under Grant 2018CFB482. The work of Song Wang was supported in part by the NSFC under Grant U1803264 and Grant 61672376. *(Corresponding author: Qin Zou.)*

Qian Wang, Zixi Li, and Lingchen Zhao are with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the State Key Laboratory of Cryptography, Beijing 100878, China (e-mail: qianwang@whu.edu.cn; lizixi0426@whu.edu.cn; lczhaocs@whu.edu.cn).

Qin Zou is with the School of Computer Science, Wuhan University, Wuhan 430072, China (e-mail: qzou@whu.edu.cn).

Song Wang is with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29201 USA, and also with the College of Intelligence and Computing, Tianjin University, Tianjin 300072, China (e-mail: songwang@cec.sc.edu).

Digital Object Identifier 10.1109/TIFS.2020.2983254

model's inputs and outputs to compromise the privacy [1], [2]. In some cases, trained models can leak information about the ownership of the training data, e.g., an adversary can perform a membership inference attack to learn whether a target data sample has been used for training the model [3], [4].

Assume that an adversary has full knowledge of the training architecture and model parameters, differential privacy has been used to solve this problem. It can ensure that the adversary cannot extract the information of any specific data from the whole dataset with high confidence, even if all the remaining data are leaked. The key idea of differential privacy is to add noise to the data for privacy protection. Several differentially private methods [5], [6] have been proposed to protect private data in transfer learning, which uses the knowledge from the source domain to improve the performance of a supervised learning algorithm over the labeled target data. For instance, Yao *et al.* [5] exploited feature-wise partitioned stacking to enhance privacy-preserving logistic regression, and combined with hypothesis transfer learning to enable learning across different organizations. However, the existing methods cannot be directly applied to supervised transfer learning to protect the privacy for the unsupervised domain adaptation. Due to the lack of labeled target data for domain adaptation, it is more difficult to obtain a good classification performance while adding noise for privacy protection.

In this paper, we propose a novel framework for deep domain adaptation with a strong privacy guarantee and low utility loss, called Differentially Private Domain Adaptation (DPDA). Specifically, the differentially private domain adaptation task mainly includes two phases. In the first phase, the model is trained by the labeled data using a traditional optimization algorithm. In the second phase, the unlabeled target data is exploited to obtain domain-invariant features by an adversarial learning strategy [7]. The shallow layers of the neural networks are trained in a fine-tuning manner, and the first few layers of them are frozen to preserve the efficacy in domain adaptation.

To achieve differential privacy with a modest utility loss, we only add noise into the specified gradients in our model. DPDA trains the model on the source and target data separately, and the noise is added only in the training process on unlabeled target data to reduce the privacy cost, since the labeled source data usually comes from public training datasets in practice. To reduce the privacy loss caused by running a large number of epochs, we use the RDP accountant [8] to track the detailed information of the privacy loss and train our model within a modest privacy budget. Furthermore, we propose a framework to protect both the source and target data for domain adaptation, called Global Differentially Private Domain Adaptation (G-DPDA). Our main contributions are three-fold.

- We propose a new and novel framework to protect sensitive private training data in domain adaptation. To our best knowledge, we are the first to provide a feasible solution for privacy-assured domain adaptation.
- We add Gaussian noise into the gradients in specific layers and phases to achieve (ϵ, δ)-differential privacy. We also use an adversarial-learning strategy to obtain domain-invariant features for the unlabeled target data classification.
- We validate the effectiveness by comparing our design with several state-of-the-art methods on two standard benchmark datasets. The results demonstrate that our scheme can perform domain adaptation tasks with a high classification accuracy for unlabeled target data while protecting individual privacy with only a modest privacy loss.

II. Related Work

A. Domain Adaptation

Domain adaptation aims to build models to decrease the difference between the probability distributions of different domains [9]. It solves the problem of lack of labeled training data. So far, many deep learning-based domain adaptation methods have been developed in the past few years [10]–[15]. Recently, unsupervised adaptation, where the target-domain data have no label or sparse labels, has been studied in many models of deep neural networks [16]–[20]. Prior methods are mainly built upon two different strategies. The first one adds the training loss with Maximum Mean Discrepancy (MMD) [21], and computes the norm of the difference between means of two domains. The MMD loss has been used in several unsupervised adaptation models, such as the Deep Domain Confusion method (DDC) [22] and the Deep Adaptation Networks (DAN) [23]. These models realize domain adaptation by minimizing the distance between domains. The second one, which utilizes a domain-adversarial loss to minimize the domain shift, is effective in unsupervised adaptation. It maximizes the loss of the feature extractor while minimizing the loss of domain classifier, and thus reducing the discrimination between domains. Many methods, e.g., the gradient reversal algorithm (ReverseGrad) [7], Deep Reconstruction-Classification Networks (DRCN) [24] and the conditional adversarial domain adaptation method [25], all follow this strategy to obtain domain-invariant features. Note that, Generative Adversarial Networks (GANs) [26] can also be used to solve the problem that the labeled training data is lacking. For example, the Coupled Generative Adversarial Networks (CoGANs) [27] achieve domain adaptation by using two GANs to generate the source-domain and target-domain data and sharing the corresponding weights.

B. Privacy-Preserving Deep Learning

Deep learning is a branch of machine learning. By embedding nonlinear functions into a multi-layer network architecture, deep learning can achieve a strong abstraction power. We are already seeing its superior performance in many application fields like speech recognition [28], image recognition [29], object detection [30], [31] and image retrieval [32] etc. It is worth noting that in some scenarios, the training data used for deep learning may be sensitive, e.g., the biomedical data about diseases [33], genetics [34] and biometrics [35], etc. A major privacy concern is that the adversary may want to determine whether a particular data is included in the input dataset [36], [37].

Differential privacy has become a common practice to prevent the adversary from inferring any information about a specific record with high confidence. Many differentially private machine learning and data analysis algorithms have been proposed, e.g., privacy-preserving boosting, logistic regression, support vector machines, neural network, empirical risk minimization, stream data processing, and parameter estimation [6], [38]–[41]. However, the existing methods only focus on supervised learning while domain adaptation considers unsupervised learning. To the best of our knowledge, the problem of how to achieve privacy-preserving domain adaptation has not been well addressed so far.

III. Background

A. Domain Adaptation

We consider a general framework for unsupervised domain adaptation [42]. Let X be the input space and $Y = \{1, \ldots, L\}$ be the set of L possible labels. We assume that there are two different distributions called the source domain $p_s(x, y)$ and the target domain $p_t(x, y)$ over $X \times Y$, respectively. A set of labeled source samples S^{N_s} are drawn from $p_s(x, y)$, and a set of unlabeled target sample T^{N_t} are drawn from $p_t(x)$, where $p_t(x)$ is the marginal distribution of $p_t(x, y)$ over X. Formally, the two datasets can be represented as

$$S^{N_s} = \{(x_i, y_i)\}_{i=1}^{N_s} \sim (p_s(x, y))^{N_s}, \tag{1}$$

$$T^{N_t} = \{(x_j)\}_{j=1}^{N_t} \sim (p_t(x))^{N_t}, \tag{2}$$

where N_s and N_t are the number of samples from the source domain and the target domain respectively. Then, the goal is to make the classifier $\psi : X \to Y$ have a low target risk

$$\mathcal{R}_{p_t(x,y)}(\psi) = \Pr_{(x,y)\sim p_t(x,y)}(\psi(x) \neq y). \tag{3}$$

We observe that minimizing domain shift by exploiting the adversarial loss is a scalable method for achieving unsupervised domain adaptation, since it achieves the state-of-the-art results and is relatively easy to tune for us. To achieve domain adaptation by adding adversarial losses, the server (the owner of the source-domain data) firstly has access to the training samples $X_s = \{x_i\}(i \in \{1, \ldots, N_s\})$ and predicts the label $y \in Y_s$ for each input $x_s \in X_s$. In the meanwhile, the server collects target-domain data from users, and trains the model with the training samples $X_{mix} = \{x_k\}(k \in \{1, \ldots, N\})$ drawn from both the source domain and the target domain based on the marginal distributions $p_s(x)$ and $p_t(x)$, where N is the number of samples in X_{mix}. In this phase, each training sample x_k has a corresponding domain label $d_k \in \{0, 1\}$. If x_k comes from the source distribution $p_s(x)$, then $d_k = 0$. If it comes from the target distribution $p_t(x)$, then $d_k = 1$.

Generally, a domain adaptation model [18] can be decomposed into three parts: *feature extractor* G_f, *label predictor* G_y, and *domain classifier* G_d (see Figure 2). Firstly, the data is mapped by a mapping G_f to a feature vector f. Then, to obtain good prediction performance on the source domain, the feature vector f is mapped by a mapping $G_y(G_f(x_i; \theta_f); \theta_y)$ with the parameters θ_y for $x_i \in X_s$ to the label y. To make the distribution $p_s(f) = G_f(x; \theta_f)|x \sim p_s(x)$ look like the $p_t(f) = G_f(x; \theta_f)|x \sim p_t(x)$, the same feature vector f is mapped by a mapping $G_d(G_f(x_k; \theta_f); \theta_d)$ with the parameters θ_d for $x_k \in X_{mix}$ to the domain label d. If these two distributions are similar, the accuracy of the label prediction on the target domain will be the same as that on the source domain. Finally, the classifier can be used to classify the target data.

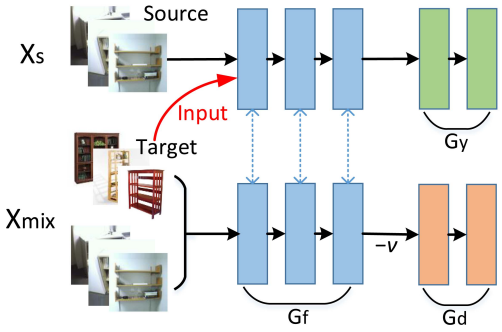


Fig. 2. An illustration of the domain adaptation model exploited to classify the unlabeled target data with the knowledge of labeled source data.

To minimize the domain shift between $p_s(f)$ and $p_t(f)$, the adversarial process [7] is introduced to enable the *feature extractor* G_f to maximize the loss of the domain classifier and the *domain classifier* G_d to minimize the loss of the domain classifier simultaneously. The function of domain adaptation is formulated as

$$\begin{aligned} E(\theta_f, \theta_y, \theta_d) &= \sum_{i=1}^{N_s} L_y\left(G_y(G_f(x_i; \theta_f); \theta_y), y_i\right) \\ &\quad - \nu \sum_{k=1}^{N} L_d\left(G_d(G_f(x_k; \theta_f); \theta_d), y_k\right) \\ &= \sum_{i=1}^{N_s} L_y^i(\theta_f, \theta_y) - \nu \sum_{k=1}^{N} L_d^k(\theta_f, \theta_d), \end{aligned} \tag{4}$$

where $L_y(\cdot, \cdot)$ is the loss for label prediction, $L_d(\cdot, \cdot)$ is the loss for domain classification, and L_y^i and L_d^k denote the loss functions at the i-th and k-th training phases, respectively. The parameter ν is used to maximize the loss of the domain classifier.

B. Differential Privacy

Nowadays, differential privacy [43] has become a powerful tool and provides strong privacy guarantees for algorithms on aggregate databases. In the domain adaptation model, for instance, the privacy guarantee is provided for the training algorithm on the source and target datasets. Moreover, differential privacy is defined in terms of pairs of adjacent databases that only differ by one training example.

Differentially private mechanisms protect the private data by adding noise on the query answers. The confidence of any inferences about individual users will be similar enough before and after the private release, so the attacker cannot infer any individual information even though all the information of other users is leaked and controlled [40].

Definition 1: A randomized mechanism $\mathcal{M}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent inputs $d(d' \in \mathcal{D})$ and for any subset of outputs $S \subseteq \mathcal{R}$, it holds that

$$Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(d') \in S] + \delta. \tag{5}$$

This definition guarantees that each output of algorithm $\mathcal{M}$ is nearly equally likely on the adjacent databases [44]. As there are two databases from different distributions in domain adaptation, differential privacy is defined separately for source domain $\mathcal{D}_{source}$ and target domain $\mathcal{D}_{target}$. In our work, each training source or target dataset is a set of image-label pairs, and the two sets are adjacent if they differ in a single entry. For instance, if one image-label pair from $\mathcal{D}_{target}$ is present in one set and absent in the other, these two sets are adjacent, and the definition can guarantee that each output of the domain adaptation model for adjacent sets from $\mathcal{D}_{target}$ is almost equally likely.

The privacy budget ϵ controls the amount of the difference between d and d'. A smaller ϵ can lead to a stronger privacy guarantee of $\mathcal{M}$. We also add a term δ introduced by [45]. δ allows for the probability that the original ϵ-differential privacy can be broken with probability δ, and it is better to have this value smaller than $1/|d|$. In addition, composability is a useful property in differential privacy, and it guarantees that if every component of a mechanism is differentially private, their composition is also differentially private. i.e., if each algorithm of $(\mathcal{M}_1, \ldots, \mathcal{M}_n)$ is (ϵ, δ)-DP, then their n-fold adaptive composition is $(n\epsilon, n\delta)$-DP.

Definition 2: For neighboring databases $d, d' \in \mathcal{D}$, a mechanism $\mathcal{M}$ and an outcome $o \in \mathcal{R}$, define the privacy loss at o as

$$c(o; \mathcal{M}, d, d') \triangleq log \frac{Pr[\mathcal{M}(d) = o]}{Pr[\mathcal{M}(d') = o]}. \tag{6}$$

Privacy loss is defined as a random variable closely related to the added random noise. It will be useful to define the privacy loss and the privacy loss random variable because the differences in the probability distribution of the result after running $\mathcal{M}$ on d and d' can be captured [44]. To specify that this random variable is mostly small, one way is to bound the Rényi divergence of $\mathcal{M}(d)$ and $\mathcal{M}(d')$.

Definition 3: Rényi Divergence [46]. The Rényi divergence of order α between the two distributions P and Q is defined as

$$D_{\alpha}(P||Q) = \frac{1}{\alpha - 1} \log \int P(o)^{\alpha} Q(o)^{1-\alpha} do. \tag{7}$$

Rényi divergence becomes the max divergence when $\alpha \to \infty$, and setting $P = \mathcal{M}(d)$ and $Q = \mathcal{M}(d')$ can ensure that $D_{\alpha}(P||Q) = \frac{1}{\alpha-1} \log \mathbb{E}_c[e^{(\alpha-1)c}]$, where c is the privacy loss variable $c(o; \mathcal{M}, d, d')$. Thus, a bound on the Rényi divergence over all orders $\alpha \in (0, \infty)$ is equivalent to $(\epsilon, 0)$-DP. Moreover, as $\alpha \to 1$, this bound approaches the expected value of $c(o; \mathcal{M}, d, d')$ equal to Kullback-Leibler divergence $KL(\mathcal{M}(d)||\mathcal{M}(d'))$ [47]. Rényi Differential Privacy is defined based on this behavior.

Definition 4: Rényi Differential Privacy (RDP) [48]. A randomized mechanism $\mathcal{M}(d)$ is said to be (α, ϵ)-Rényi differentially private if and only if its distribution over any two adjacent inputs d and d' satisfies

$$D_{\alpha}(\mathcal{M}(d)||\mathcal{M}(d')) \leq \epsilon. \tag{8}$$

Here, α is chosen in RDP to tune the amount of concern placed on unlikely large values of $c(o; \mathcal{M}, d, d')$ versus the average value of $c(o; \mathcal{M}, d, d')$.

C. Deep Learning With Differential Privacy

Generally, a deep neural network is composed of multiple layers with hidden neurons and nonlinear functions. Given a set of data examples, the goal of the learning process of the network is to find a set of parameters θ that optimize the output close enough to the ground truth. More precisely, a loss function $\mathcal{L}$ is defined to represent the penalty for mismatching the training data. The loss $\mathcal{L}(\theta) = \frac{1}{N}\mathcal{L}(\theta, x_i)$ is the mean value of the loss over the training examples $x_1, \ldots, x_N$. To achieve the goal of learning, the loss should be small enough by the specific θ after training with an effective optimization. Mini-batch stochastic gradient descent (SGD) algorithm [49] has been proposed to complete this task by minimizing the cross-entropy error for the model outputs and the corresponding labels. In this algorithm, a batch B of random examples is extracted, and $\mathbf{g}_B = 1/|B| \sum_{x \in B} \nabla_{\theta}\mathcal{L}(\theta, x)$ is computed to estimate the gradient $\nabla_{\theta}\mathcal{L}(\theta)$ at each step in this algorithm. Then the parameters θ are updated with the gradient direction $\mathbf{g}_B$.

The method proposed by [40] adds Gaussian noise into the gradients to protect private data. It takes a random sample $\mathcal{L}_t$ at every training step t and computes the gradient $\mathbf{g}_t(x_i)$ for each tuple $x_i \in \mathcal{L}_t$. Then it clips each gradient in l_2 norm and adds Gaussian noise into them. Finally, the differentially private parameters θ learned by the algorithm can be published without privacy leakages.

IV. Differentially Private Domain Adaptation Model

A. Problem Statement

There are two data distributions (or domains): one is the source distribution $p_s(x, y)$, and the other is the target distribution $p_t(x)$. These two data distributions are similar but have difference. The source-domain data $S^{N_s} = \{(x_i, y_i)\}_{i=1}^{N_s}$ are adequate and labeled, and the target-domain data $T^{N_t} = \{(x_j)\}_{j=1}^{N_t}$ are unlabeled. Our ultimate goal is to train a model which can predict the labels of the target-domain data while preventing the privacy leakages of the sensitive training data.

Since the distributions of the source-domain data and the target-domain data are usually different, the model only trained by the source-domain data will have poor performance in classifying the target-domain data. Therefore, exploiting both the labeled source-domain data and the unlabeled target-domain data to train the classification model is essential for domain adaptation. Because an adversary who can use the model arbitrarily may be able to extract partial information of the training data, we add differentially private noise into the training process to protect the sensitive information.

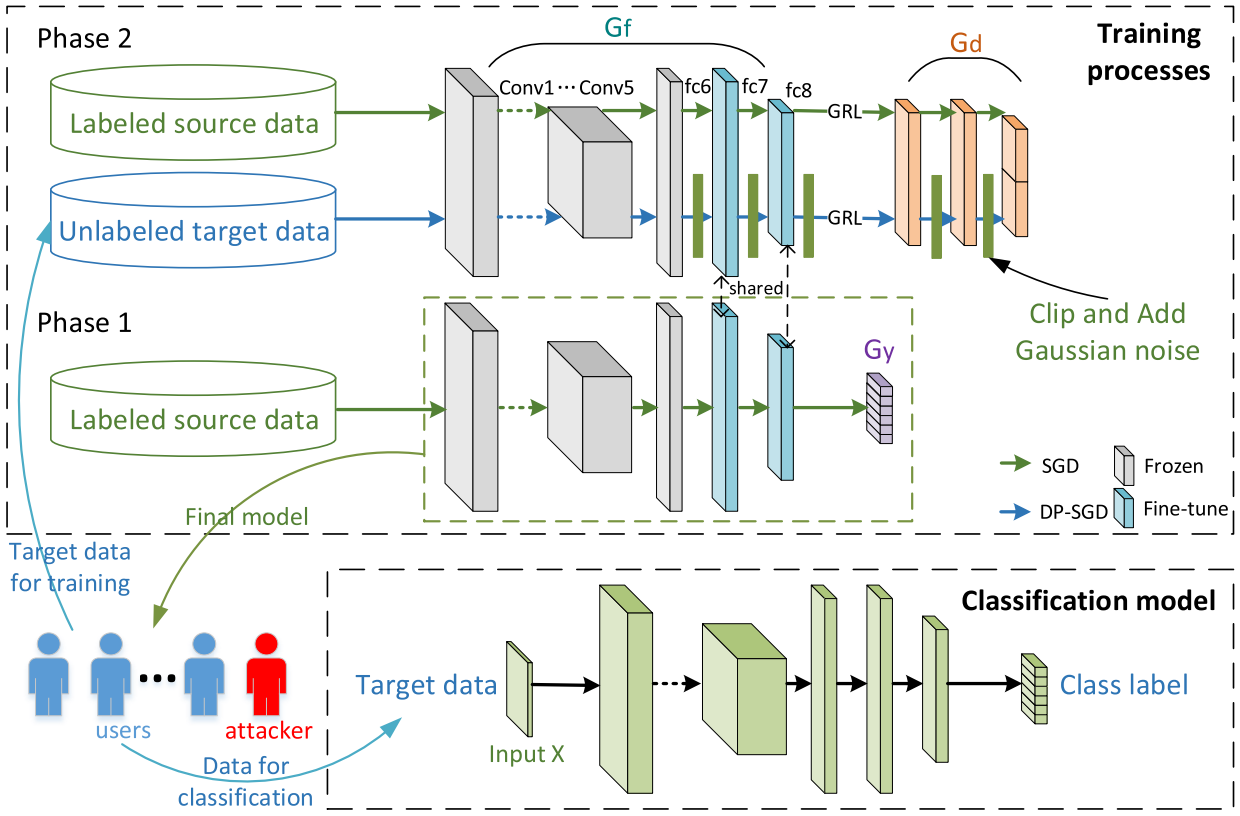


Fig. 3. The proposed architecture includes two phases and three main parts. In Phase 1, the training algorithm minimizes the label prediction loss in the *feature extractor* (blue) and the *label predictor* (purple) by running the standard SGD algorithm on the labeled data. In Phase 2, the training algorithm maximizes the domain classifier loss in the *feature extractor* and minimizes the domain classifier loss in the *domain classifier* (orange). The model is trained by running the differentially private SGD algorithm on the unlabeled target data and the standard SGD algorithm on the source data in Phase 2. The first 6 layers are frozen and we only finetune $fc7$ and $fc8$ in the *feature extractor*. The weights in both two phases are shared in $fc7$ and $fc8$.

B. Network Architecture

To prevent privacy leakages of the target data from the domain adaptation model, we design a Differentially Private Domain Adaptation (DPDA) framework. Assume that the training server has access to both the source and the target data, as illustrated in Figure 3, we define a deep feed-forward architecture including two phases. In Phase 1, the server predicts the labels of source-domain training samples X_s collected based on the marginal distribution $p_s(x)$. In Phase 2, it collects target-domain training samples $X_t = \{x_j\}(j \in \{1, \ldots, N_t\})$ from users based on the marginal distributions $p_t(x)$, and trains the model with the training set X_{mix} that consists of both the source-domain and target-domain samples. Every training sample has an additional label of the domain it belongs to. Next, we present the designs of G_f, G_y, and G_d respectively.

a) Feature extractor G_f: In DPDA, the feature extractor is used to produce similar $p_s(x)$ and $p_t(x)$ over the representation space, which enables the model trained by the labeled source data to classify the unlabeled target data with better performance. The data in both the two phases are mapped by a mapping G_f. We use the AlexNet architecture [30] where five convolutional layers ($conv1 - conv5$) and three fully connected layers ($fc6 - fc8$) are contained in the feature mapping process. As the convolutional layers will learn the generic features [50], parameters on these layers can be highly adaptive across the datasets. Hence, we freeze the $conv1 - conv5$ and $fc6$ to adapt the pre-trained AlexNet to our model, and only retrain the layers $fc7$ and $fc8$. We denote the parameter vector of $fc7$ and $fc8$ in this mapping as θ_f, i.e., $f = G_f(x; \theta_f)$, and share these parameters between the two phases. Then the feature vector f is mapped by a mapping $G_y(G_f(x_i; \theta_f); \theta_y)$ with the parameters θ_y for $x_i \in X_s$ to the label y in Phase 1, and a mapping $G_d(G_f(x_k; \theta_f); \theta_d)$ with the parameters θ_d for $x_k \in X_{mix}$ to the label d in Phase 2.

b) Label predictor G_y: The label predictor uses the labeled source data to train a classification model. As our goal is to achieve high performance on the source domain in Phase 1, we optimize G_f and G_y to minimize the empirical loss for the source-domain samples. The label prediction loss is

$$E(\theta_f, \theta_y) = \sum_{i=1}^{N_s} L_y\left(G_y(G_f(x_i; \theta_f); \theta_y), y_i\right) = \sum_{i=1}^{N_s} L_y^i(\theta_f, \theta_y). \quad (9)$$

Since this phase is a standard feed-forward operation in deep learning, we use the standard SGD to update the parameters.

c) Domain classifier G_d: As the private target data is used to train the domain classifier, G_d needs to protect the

private information of the target data while minimizing the domain shift in DPDA. The feature vector f is mapped by a mapping $G_d(G_f(x;\theta_f);\theta_d)$ with the parameters θ_{ds} for $x_i \in X_s$ or the parameters θ_{dt} for $x_j \in X_t$ to the domain label d in Phase 2. The loss of the *domain classifier* G_d corresponding to the discrimination between the two feature distributions are exploited to make the distribution $p_s(f)$ similar to $p_t(f)$. In the training procedure, we introduce the adversarial process [7] to let G_f maximize the loss of the domain classifier and G_d minimize the loss of the domain classifier simultaneously. As traditional domain adaptation methods train the models by the mixed data X_{mix} directly to minimize the domain shift, it is difficult to solely protect the privacy for the target data. In our scheme, we train the model by the source data X_s and the target data X_t separately. Therefore, the domain prediction loss can be transformed into the following form

$$\begin{aligned} E(\theta_f,\theta_{ds},\theta_{dt}) &= -\nu\sum_{i=1}^{N_s} L_d\left(G_d(G_f(x_i;\theta_f);\theta_{ds}),y_i\right) \\ &\quad -\nu\sum_{j=1}^{N_t} L_d\left(G_d(G_f(x_j;\theta_f);\theta_{dt}),y_j\right) \\ &= -\nu\left(\sum_{i=1}^{N_s} L_d^i(\theta_f,\theta_{ds})+\sum_{j=1}^{N_t} L_d^j(\theta_f,\theta_{dt})\right). \end{aligned} \tag{10}$$

After iterative training in the two phases with the parameters shared to layers $fc7$ and $fc8$, the distributions of the source domain and the target domain will be similar such that we can use the label predictor to classify the target data.

Based on the method proposed in [40], we inject Gaussian noise into the gradients to achieve differential privacy and protect the private training data. Since we train the model by the source and target data separately in DPDA, the noise can be only added to the training on the target data, which reduces the utility loss caused by the noise as much as possible. Specifically, we compute the gradients of a random subset of the target-domain examples with size M and add noise to them at each step of SGD (see details in Section IV-C). Finally, this model can be used by any individual, and its characteristics are not specific to any single individual.

C. Optimization in Two Different Phases

In this section, we show how to optimize the two phases. In Phase 1, we use the standard SGD to minimize the empirical loss function $E(\theta_f,\theta_y)$. Parameters θ_f and θ_y are updated by the following stochastic updates

$$\theta_f \longleftarrow \theta_f - \eta\frac{\partial L_y^i}{\partial\theta_f}, \tag{11}$$

$$\theta_y \longleftarrow \theta_y - \eta\frac{\partial L_y^i}{\partial\theta_y}, \tag{12}$$

where η is the learning rate.

As discussed above, to reduce the utility loss caused by differential privacy, we only add noise into gradients in Phase 2 as no target-domain data is used in Phase 1. In each iteration of the training on X_s, we take a batch of examples M_s from X_s. As we do not have to protect the public source data X_s in DPDA, only the standard SGD is used to minimize the empirical loss function $E(\theta_f,\theta_{ds})$. For the training on the target data, we first take a batch of examples M_t from X_t, and clip the gradients in l_2 norm to bound the influence of each individual example to achieve differential privacy, i.e., for a predefined threshold C, the gradient vector $\mathbf{g}/\max(1,\|\mathbf{g}\|_2/C)$ replaces the gradient vector $\mathbf{g}$. Then we perturb these clipped gradients by the Gaussian mechanism. In Phase 2, parameters θ_{ds}, θ_{dt} and θ_f are updated by the following stochastic updates

$$\theta_{ds} \longleftarrow \theta_{ds} - \eta\frac{1}{M}\sum_i\frac{\partial L_d^i}{\partial\theta_{ds}}, \tag{13}$$

$$\begin{aligned} \theta_{dt} \longleftarrow \theta_{dt} &- \eta\frac{1}{M}\sum_j\Big(\mathcal{N}(0,\sigma^2\ C^2\mathbf{I}) \\ &+\frac{\partial L_d^j}{\partial\theta_{dt}}/\max\Big(1,\frac{\|\frac{\partial L_d^j}{\partial\theta_{dt}}\|_2}{C}\Big)\Big), \end{aligned} \tag{14}$$

$$\begin{aligned} \theta_f \longleftarrow \theta_f &- \eta\frac{1}{M}\Bigg(-\nu\sum_j\Big(\mathcal{N}(0,\sigma^2C^2\mathbf{I}) \\ &+\frac{\partial L_d^j}{\partial\theta_f}/\max\Big(1,\frac{\|\frac{\partial L_d^j}{\partial\theta_f}\|_2}{C}\Big)\Big) - \nu\sum_i\frac{\partial L_d^i}{\partial\theta_f}\Bigg), \end{aligned} \tag{15}$$

where η is the learning rate, M is the batch size, C is the gradient norm bound, σ is the noise scale, and $\mathcal{N}(0,\sigma^2\ C^2\mathbf{I})$ is the normal Gaussian distribution.

However, in Phase 2, we cannot directly invoke the standard SGD algorithm because there is a $-\nu$ factor in the empirical loss function $E(\theta_f,\theta_d)$, which plays an important role in maximizing the loss of domain classifier in the *feature extractor* G_f. As a result, we add the *gradient reversal layer* (GRL) proposed by [7] to make SGD available for this phase. During the forward propagation, GRL just copies the data without changing it. But in the backpropagation, GRL multiplies the gradient by $-\nu$. The gradient reversal layer is defined as $R_\nu(x)$

$$R_\nu(x) = x, \tag{16}$$

$$\frac{dR_\nu}{dx} = -\nu\mathbf{I}, \tag{17}$$

where $\mathbf{I}$ is an identity matrix. Then domain the prediction loss can be defined as

$$\begin{aligned} \tilde{E}(\theta_f,\theta_{ds},\theta_{dt}) &= \sum_{i=1}^{N_s} L_d\left(G_d(R_\nu(G_f(x_i;\theta_f));\theta_{ds}),y_i\right) \\ &\quad+\sum_{j=1}^{N_t} L_d\left(G_d(R_\nu(G_f(x_j;\theta_f));\theta_{dt}),y_j\right). \end{aligned} \tag{18}$$

Then we can use the standard SGD algorithm to optimize the domain classifier.

After running updates Eqs. (13)-(15) for Eq. (10), $p_t(f)$, which is perturbed by noise and $p_s(f)$, will be similar enough. Therefore, after updating the loss in Eq. (9) according to

Eqs. (11) and (12), the unlabeled target data can be classified with higher accuracy by the label predictor $y(x) = G_y(G_f(x;\theta_f);\theta_y)$. The outline of our basic method for training a differentially private domain adaptation model is presented in Algorithm 1.

Algorithm 1 Differentially Private Domain Adaptation

Input: Examples $X_s = \{x_i\}, i \in \{1,\ldots,N_s\}$ from the source domain and $X_t = \{x_j\}, j \in \{1,\ldots,N_t\}$ from target domain, loss function $E(\theta_f,\theta_y)$ and $\tilde{E}(\theta_f,\theta_{ds},\theta_{dt})$. Parameters: learning rate η, noise level σ, batch size M, gradient norm bound C.

Output: Model parameters $\theta_{ds}, \theta_{dt}, \theta_y$, and the overall privacy cost (ϵ,δ).

Initialize θ_{ds}, θ_{dt}, θ_y randomly

for $t \in [T]$ **do**

(1) Updata parameters θ_y in Phase 1:

Take random samples M_s with probability M/N_s

For each $i \in M_s$, compute $\mathbf{g}_y^t(x_i) \leftarrow \frac{\partial L_y^i}{\partial \theta_y^t}$

$\tilde{\mathbf{g}}_y^t \leftarrow \frac{1}{M}\sum_i \tilde{\mathbf{g}}_y^t(x_i)$

$\theta_y^{t+1} \leftarrow \theta_y^t - \eta\tilde{\mathbf{g}}_y^t$

(2) Updata parameters θ_{dt} in Phase 2:

Take random samples M_t with probability M/N_t

For each $j \in M_t$, compute $\mathbf{g}_{dt}^t(x_j) \leftarrow \frac{\partial L_d^j}{\partial \theta_{dt}^t}$

Clip gradient

$\bar{\mathbf{g}}_{dt}^t(x_j) \leftarrow \mathbf{g}_{dt}^t(x_j)/\max\left(1, \frac{\|\mathbf{g}_{dt}^t(x_j)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_{dt}^t \leftarrow \frac{1}{M}\left(\sum_j \bar{\mathbf{g}}_{dt}^t(x_j) + \mathcal{N}(0,\sigma^2\ C^2\mathbf{I})\right)$

$\theta_{dt}^{t+1} \leftarrow \theta_{dt}^t - \eta\tilde{\mathbf{g}}_{dt}^t$

(3) Updata parameters θ_{ds} in Phase 2:

Take random samples M_s with probability M/N_s

For each $i \in M_s$, compute $\mathbf{g}_{ds}^t(x_i) \leftarrow \frac{\partial L_d^i}{\partial \theta_{ds}^t}$

$\tilde{\mathbf{g}}_{ds}^t \leftarrow \frac{1}{M}\sum_i \bar{\mathbf{g}}_{ds}^t(x_i)$

$\theta_{ds}^{t+1} \leftarrow \theta_{ds}^t - \eta\tilde{\mathbf{g}}_{ds}^t$

end for

D. Global Differentially Private Domain Adaptation

In the DPDA model, as the private target data is not exploited in Phase 1, we can only add noise in Phase 2. This will help to improve the model accuracy and reduce the computation cost. Because the source data is always published, we first focus on preserving the target data privacy in DPDA. In this section, we discuss how to protect the two types of data simultaneously and propose the Global Differentially Private Domain Adaptation (G-DPDA) method, which achieves differential privacy in both phases.

As shown in Figure 4, to protect the source-domain data used in Phase 1, we need to add noise to the whole training process. Specifically, we add noise in the same way as shown in Section IV-C, and all the parameters are updated by the differentially private SGD algorithm. In Phase 1, we take a batch of samples M_s collected from the source distribution to compute the gradients. Then the gradients are clipped and

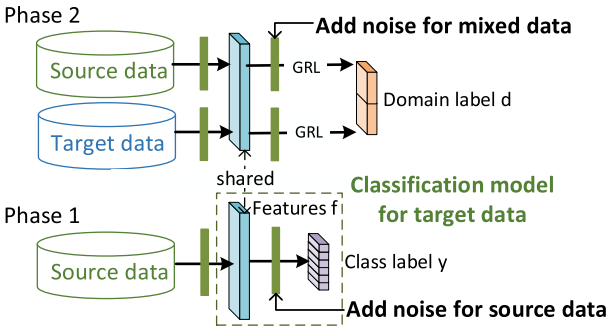


Fig. 4. Global differentially private domain adaptation in shallow neural networks.

perturbed as follows

$$\theta_y \longleftarrow \theta_y - \eta\frac{1}{M}\sum_i\Big(\mathcal{N}(0,\sigma^2\ C^2\mathbf{I}) + \frac{\partial L_y^i}{\partial\theta_y}/\max\Big(1,\frac{\|\frac{\partial L_y^i}{\partial\theta_y}\|_2}{C}\Big)\Big), \tag{19}$$

$$\theta_f \longleftarrow \theta_f - \eta\frac{1}{M}\sum_i\Big(\mathcal{N}(0,\sigma^2C^2\mathbf{I}) + \frac{\partial L_y^i}{\partial\theta_f}/\max\Big(1,\frac{\|\frac{\partial L_y^i}{\partial\theta_f}\|_2}{C}\Big)\Big). \tag{20}$$

In Phase 2, we add noise to gradients in the training processes for the source data and the target data separately. The model is trained by a batch of samples M_s from X_s and a batch of samples M_t from X_t in each epoch. For each $x_i \in M_s$ and $x_j \in M_t$, the parameters θ_{dt} are updated in the same way as Eq. (14), while θ_{ds} and θ_f are perturbed by the following stochastic updates

$$\theta_{ds} \longleftarrow \theta_{ds} - \eta\frac{1}{M}\sum_i\Big(\mathcal{N}(0,\sigma^2\ C^2\mathbf{I}) + \frac{\partial L_d^i}{\partial\theta_{ds}}/\max\Big(1,\frac{\|\frac{\partial L_d^i}{\partial\theta_{ds}}\|_2}{C}\Big)\Big), \tag{21}$$

$$\theta_f \longleftarrow \theta_f - \eta\frac{1}{M}\Bigg(-\nu\sum_i\Big(\mathcal{N}(0,\sigma^2\ C^2\mathbf{I}) + \frac{\partial L_d^i}{\partial\theta_f}/\max\Big(1,\frac{\|\frac{\partial L_d^i}{\partial\theta_f}\|_2}{C}\Big)\Big) - \nu\sum_j\Big(\mathcal{N}(0,\sigma^2C^2\mathbf{I}) + \frac{\partial L_d^j}{\partial\theta_f}/\max\Big(1,\frac{\|\frac{\partial L_d^j}{\partial\theta_f}\|_2}{C}\Big)\Big)\Bigg). \tag{22}$$

After the iterative training process perturbed by Gaussian noise, the trained model will not expose any sensitive information about the whole training dataset, i.e., both the source data and the target data are well protected.

E. Privacy Analysis

In this section, we present the formal privacy analysis of DPDA and G-DAPA. Recall that the goal of DPDA is to

protect the privacy of individual target-domain data. We will first prove that Algorithm 1 can achieve $(\epsilon + \frac{\log 1/\delta}{\alpha-1}, \delta)$-differential privacy while setting appropriate values for the noise scale and the batch size.

If we directly use the basic composition theorem to quantify the privacy loss, the value of ϵ will be extremely large after running multiple epochs. Therefore, we choose to use RDP to track the privacy loss. Let q be the sampling probability M/N_t, the following main theorem of the RDP accountant has been proved in [8].

Theorem 1: If $q \leq \frac{1}{5}$, $\sigma \geq 4$, and α satisfies

$$1 < \alpha \leq \frac{1}{2}\sigma^2 L - 2\ln\sigma, \tag{23}$$

$$\alpha \leq \frac{\frac{1}{2}\sigma^2 L^2 - \ln 5 - 2\ln\sigma}{L + \ln(q\alpha) + 1/(2\sigma^2)}, \tag{24}$$

where $L = \ln\left(1 + \frac{1}{q(\alpha-1)}\right)$, then SGM applied to a function of ℓ_2-sensitivity 1 satisfies (α, ϵ)-RDP where

$$\epsilon \triangleq 2q^2\alpha/\sigma^2. \tag{25}$$

As an (α, ϵ)-RDP implies $(\epsilon_\delta, \delta)$-differential privacy for any given probability $\delta > 0$, we use the following proposition proved in [48] to obtain $(\epsilon_\delta, \delta)$-differential privacy from (α, ϵ)-RDP.

Proposition 2: (From RDP to (ϵ, δ)-DP). If $\mathcal{M}$ is an (α, ϵ)-RDP mechanism, it also satisfies $(\epsilon + \frac{\log 1/\delta}{\alpha-1}, \delta)$-differential privacy for any $0 < \delta < 1$.

Combining the above results, we can achieve $(\epsilon + \frac{\log 1/\delta}{\alpha-1}, \delta)$-differential privacy in protecting the target-domain data while using the RDP accountant to trace the privacy cost. As stated in Section III-B, both ϵ and δ should be small in order to enforce a strong privacy guarantee. Therefore, we use the convention that $\delta = 10^{-5}$ in our scheme, and can get a small ϵ by using RDP accountant.

In the G-DPDA model, both the source and target datasets are protected by adding noise but with different privacy budgets. Specifically, G-DPDA achieves $(\epsilon + \frac{\log 1/\delta}{\alpha-1}, \delta)$-differential privacy for the source-domain data in the two training processes respectively when setting q as M/N_s. Thanks to the composability of differential privacy, G-DPDA finally achieves $(2(\epsilon + \frac{\log 1/\delta}{\alpha-1}), 2\delta)$-differential privacy for the source data. For target-domain data, G-DPDA still achieves $(\epsilon + \frac{\log 1/\delta}{\alpha-1}, \delta)$-differential privacy, and the proving process is just the same as the process in DPDA. Based on the definition of differential privacy, the attacker cannot extract the information about the private training samples by observing the outputs of the model even if he/she knows all the other samples in the training dataset.

V. Experiments

In this section, we implement the proposed DPDA and G-DPDA frameworks by TensorFlow, and evaluate them on two popular domain adaptation benchmark datasets: the *Office-31* [51] and the *Amazon review dataset* [52], which are used in some representative works about domain adaptation such as [53], [54].

We run the experiments as follows. First, we evaluate the DPDA method introduced in Section IV-C. Then, we evaluate the G-DPDA method proposed in Section IV-D. Finally, we evaluate the different training strategies by changing the number of frozen layers in the model. We compare our proposed methods with some representative domain adaptation and deep learning methods.

A. Experiments With Deep Neural Networks

1) Office-31 Dataset: This dataset, which consists of 4,110 images with 31 classes collected from three different domains: *Amazon* (A), *Webcam* (W) and *DSLR* (D), and is a popular benchmark in domain adaptation tasks. The images in A are downloaded from amazon.com, the images in W are taken by web cameras, and the images in D are taken by digital SLR cameras. We present several representative samples in Figure 5. The 31 classes consist of the common objects in office settings, such as chairs, laptops, and bicycles.

Images from the target domain are split into two parts for test and training respectively. As the number of images in the *Office-31* is not very large, we finetune our model by using a CNN pre-trained on the ImageNet. We train the model by the source-domain data and unlabeled target-domain data, and test the model by the labeled target-domain data. We evaluate the performance of the trained model on three domain adaptation tasks ($A \rightarrow W$, $D \rightarrow W$ and $W \rightarrow D$) which are widely adopted in related research works.

Seven benchmarks are included in the comparisons: Convolutional Neural Networks (CNN) [30], Geodesic Flow Kernel (GFK) [53], Transfer Component Analysis (TCA) [42], Deep Correlation Alignment (D-CORAL) [55], Deep Domain Confusion (DDC) [22], Deep Adaptation Network (DAN) [23] and Domain Adversarial Neural Network (DANN) [18]. Except for CNN, all the other six methods perform the domain adaptation tasks. Specifically, GFK and TCA project the distributions of the source and target domains into a lower-dimensional manifold to achieve domain adaptation. DDC learns a discriminative and domain-invariant representation by adding a single linear kernel to one layer to minimize the Maximum Mean Discrepancy (MMD). DAN improves the DDC method by minimizing MMD with multiple kernels in multiple layers and inserting them into a reproducing kernel Hilbert space. D-CORAL proposes a new loss (CORAL loss) to minimize the difference between the source and target domains. DANN confuses the two domains by adding a gradient reversal layer to learn the features combining discrimination and domain-invariance.

2) Network Architectures: In general, we compose the feature extractor with five convolutional layers and three fully connected layers, which are the same as AlexNet. For the domain classifier, we stick to 3 fully connected layers ($x \rightarrow 1024 \rightarrow 1024 \rightarrow 2$). For the label classifier, we stick to only 1 fully connected layer ($x \rightarrow 31$). We use the logistic regression loss L_y and the binomial cross-entropy L_d as the loss functions for the label prediction and the domain classifier respectively.

3) Parameter Settings: In this experiment, the adaptation parameter ν is gradually changed from 0 to 1 on a

Fig. 5. Samples from the *Office-31* dataset. Left, middle and right columns show some samples from the Webcam, DSLR, and Amazon, respectively.

logarithmic scale. The learning rate η is fixed at 10^{-3} in both of the two phases. Some parameters in differential privacy are set as follows.

a) Clipping bound C: If we set the clipping bound of gradients as a small value, the true gradients may change a lot. In the meanwhile, a large clipping parameter will lead to large noise which may affect the accuracy. Empirically, we set up the clipping parameter as 3 to achieve an apposite trade-off between utility and privacy.

b) Noise level σ: The noise level in differentially private SGD algorithm might be the most important parameter. If we choose a relatively large noise level, we can run more epochs, which may improve the accuracy but decrease the privacy guarantee. By doing the comparative experiments, we choose the noise level to be 6 considering the balance between the accuracy and the privacy guarantee level.

c) Batch size M: We find that the batch size has a great influence on the test accuracy. A large batch size can improve the accuracy, but more noise will be added. Empirically, we set up the batch size as 32 to achieve a higher accuracy.

According to RDP account [8], the overall privacy loss (ϵ, δ) can be quantified by the noise level σ, the sampling ratio q and the number of epochs E. We set $\sigma = 6$ and $\delta = 10^{-5}$ in our experiments, and compute the value of ϵ as a function of the training epochs E. In order to enforce a strong privacy guarantee, we always keep ϵ and δ small. The value of ϵ is dropped from 4 to 1, and δ is dropped from 10^{-2} to 10^{-5} in the contrast experiments.

4) Different Training Strategies: Since the generic features learned in convolutional layers tend to be transferable in layers $conv1-conv5$ [50], the transferability gap between the convolutional layers in two phases may be small. On the other hand, the more layers are trained, the more noise will be injected. Hence, we opt to freeze the convolutional layers and parts of the fully connected layers in the model to avoid affecting the efficacy of fragile co-adaptation [56] when adapting the pre-trained AlexNet. In AlexNet, the last layer of the network transforms the deep features from general into specific. When transferring the higher layers $fc6-fc8$, the transferability gap becomes especially large as the domain discrepancy increases. To study the optimal scheme of retraining the higher layers, we evaluate several variants of DPDA and DANN:

- DPDA_8 and DANN_8: freeze $conv1-conv5$ and $fc6$, $fc7$, retrain $fc8$ and train other layers, respectively. The parameters of the frozen layers are copied from the pre-trained model.
- $\text{DPDA}_{7,8}$ and $\text{DANN}_{7,8}$: freeze $conv1-conv5$, $fc6$, retrain $fc7$, $fc8$ and train other layers, respectively.
- $\text{DPDA}_{6,7,8}$ and $\text{DANN}_{6,7,8}$: freeze $conv1-conv5$, retrain $fc6-fc8$ and train other layers, respectively.

5) Results and Discussions: Table I shows the performance of the unsupervised adaptation methods on the three *Office-31* domain adaptation tasks. We can observe that DPDA outperforms CNN (non-adapted) and some other classical methods while achieving $(8, 10^{-5})$-differential privacy. In Figure 6(a), we visualize and compare the test accuracy on the $A \rightarrow W$ domain adaptation task by DPDA, DANN, and CNN respectively. We can see that DPDA obtains a better classification performance than CNN while achieving $(4, 10^{-5})$-differential privacy, indicating that our DPDA model can still complete the domain adaptation tasks even if the noise is added to the gradients.

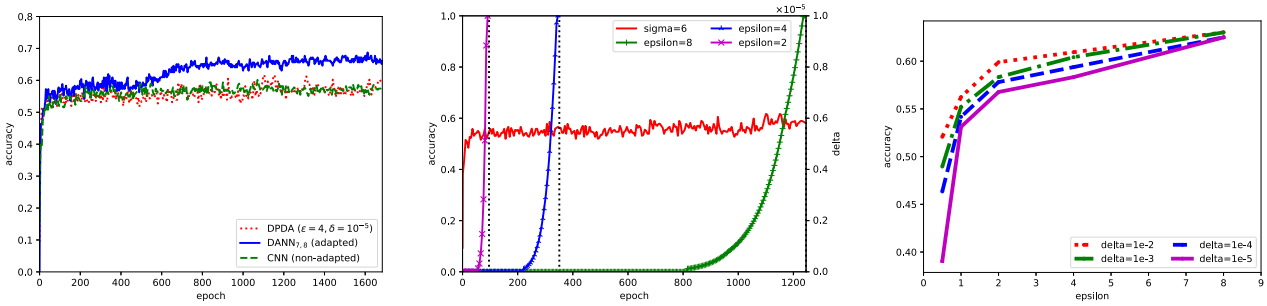
Multiple factors such as the number of training layers and the privacy budget determine the accuracy directly and need to be carefully tuned to get the optimal performance. From the experiments, we respectively observe the following factors which affect the classification accuracy for unlabeled target data in DPDA.

a) Specific parameters of noise: We conduct several contrast experiments to get a better understanding of privacy protection in DPDA. Table II shows the test accuracy for different privacy budgets ϵ on three different domain adaptation tasks, while keeping $\delta = 10^{-5}$ and $\sigma = 6$. We observe that running more epochs can achieve a higher accuracy and spend more privacy budgets. In all the three tasks, we achieve a higher accuracy with ϵ being 4 than 1. In Figure 6(b), we visualize the test accuracy with different privacy budgets

TABLE I

CLASSIFICATION ACCURACY ON THE OFFICE-31 DATASET WITH STANDARD UNSUPERVISED ADAPTATION PROTOCOL

Method	A → W	D → W	W → D	Average
CNN [30]	0.602 ± 0.4	0.930 ± 0.3	0.961 ± 0.2	0.831
GFK [53]	0.547 ± 0.0	0.921 ± 0.0	0.962 ± 0.0	0.810
TCA [42]	0.455 ± 0.0	0.811 ± 0.0	0.922 ± 0.0	0.729
DDC [22]	0.618 ± 0.4	0.950 ± 0.5	0.985 ± 0.4	0.851
DAN [23]	0.685 ± 0.4	0.960 ± 0.1	0.990 ± 0.1	0.878
D-CORAL [55]	0.664 ± 0.4	0.957 ± 0.3	0.992 ± 0.1	0.871
DANN [18]	0.668 ± 0.2	0.935 ± 0.8	0.988 ± 0.1	0.864
DPDA (Ours)	0.625 ± 0.3	0.945 ± 0.2	0.977 ± 0.0	0.849



(a) (b) (c)

Fig. 6. Test accuracy on the unsupervised *Amazon* → *Webcam* task. In all the experiments, we set the noise level σ as 6, the clipping threshold C as 3 and the batch size as 32. (a) shows the results obtained by DPDA, DANN$_{7,8}$ and CNN. (b) shows the test accuracy with different ϵ values in DPDA. (c) shows the accuracy of different (ϵ, δ) pairs and different curves correspond to varied delta values.

TABLE II

CLASSIFICATION ACCURACY ON THE OFFICE-31 DATASET FOR DIFFERENT VALUES OF EPSILON

Domain					**Accuracy**		
Source	Target	Epsilon ϵ	Delta δ	Epoch E	DPDA(Ours)	Non-Private	Non-Adapted
Amazon	Webcam	4.0	10^{-5}	353	0.617	0.708	0.602
Amazon	Webcam	2.0	10^{-5}	94	0.586	0.708	0.602
Amazon	Webcam	1.0	10^{-5}	24	0.559	0.708	0.602
DSLR	Webcam	4.0	10^{-5}	241	0.945	0.974	0.930
DSLR	Webcam	2.0	10^{-5}	64	0.936	0.974	0.930
DSLR	Webcam	1.0	10^{-5}	17	0.914	0.974	0.930
Webcam	DSLR	4.0	10^{-5}	241	0.969	0.996	0.961
Webcam	DSLR	2.0	10^{-5}	64	0.961	0.996	0.961
Webcam	DSLR	1.0	10^{-5}	17	0.945	0.996	0.961

for domain shift $A \rightarrow W$. Note that, a smaller privacy budget ϵ allows fewer epochs of training but provides a stronger privacy guarantee.

Table III presents the test accuracy for different (ϵ, δ) pairs and noise level σ in domain shift $A \rightarrow W$. We can observe that a larger σ allows more epochs of training before reaching the privacy limit. We adaptively choose parameter $\sigma = 6$ by conducting the contrast experiments. We are also able to observe that the test accuracy becomes higher when the value of ϵ drops from 4 to 1 and becomes lower when δ drops from 10^{-2} to 10^{-5}. Figure 6(c) shows the test accuracy for different (ϵ, δ) pairs. We can find that the parameter ϵ has a much larger impact on the accuracy than δ.

b) Number of training layers: Table IV presents the accuracies of four variants of DANN. We can observe that DANN$_{7,8}$ has better performance than DANN (retraining all layers in AlexNet), DANN$_8$ and DANN$_{6,7,8}$ in three domain adaptation tasks. Figure 7(a) plots the results of the three variants on the unsupervised $A \rightarrow W$ task. In Figure 7(b), we present the accuracy of the three variants of our DPDA method on the same task while setting σ as 4 and the clipping threshold as 3. Both the two figures show that only retraining both $fc7$ and $fc8$ in AlexNet can achieve the best results for domain adaptation tasks.

B. Experiments With Shallow Neural Networks

1) Amazon Review Dataset: This dataset contains 340,000 online reviews of 25 different products collected on Amazon.com. Every review is originally recorded by unigram and bigram, and we encode them into a 5,000-dimensional

TABLE III

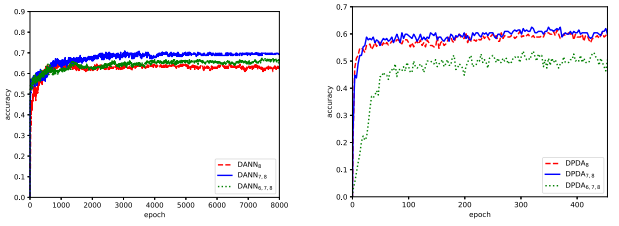
CLASSIFICATION ACCURACY AND EPOCHS ON THE AMAZON → WEBCAM TASK FOR DIFFERENT (ϵ, δ) PAIRS AND NOISE LEVEL

ϵ	1.0	2.0	4.0	4.0	4.0	4.0
δ	10^{-5}	10^{-5}	10^{-5}	10^{-4}	10^{-3}	10^{-2}
Epochs E ($\sigma = 4$)	10	40	153	185	235	321
Testing accuracy ($\sigma = 4$)	0.508	0.551	0.590	0.605	0.605	0.609
Epochs E ($\sigma = 6$)	24	94	353	428	540	736
Testing accuracy ($\sigma = 6$)	0.559	0.586	0.617	0.617	0.617	0.625
Epochs E ($\sigma = 8$)	44	169	633	765	969	1320
Testing accuracy ($\sigma = 8$)	0.566	0.594	0.594	0.594	0.594	0.602

TABLE IV

CLASSIFICATION ACCURACY ON THE OFFICE-31 DATASET OBTAINED BY FOUR VARIANTS OF DANN [18]

Method	A → W	D → W	W → D	Average
DANN [18]	0.668	0.935	0.988	0.864
DANN_8	0.672	0.951	0.989	0.871
$\text{DANN}_{7,8}$	**0.708**	**0.974**	**0.996**	**0.893**
$\text{DANN}_{6,7,8}$	0.685	0.965	0.993	0.881



(a) (b)

Fig. 7. Test accuracy for the variants of DANN [18] and DPDA on the $A \rightarrow W$ domain adaptation task. (a) and (b) show the results obtained by DANN and DPDA when training on different fully connected layers.

feature vector by using the bag-of-word method. We aim to predict positive (4 stars or more) or negative (3 stars or less) notation of reviews in this dataset. The classification label is '0' if the product is ranked 3 or fewer stars, and the label is '1' if the product is ranked 4 or more stars. If someone wants to predict the reviews of a product with no label from labeled reviews of a different product, he/she may need to use domain adaptation methods to train the prediction model because different kinds of products can be qualified by different words.

Twelve domain adaptation tasks are considered for four different types of products: *books*, *DVDs*, *electronics* and *kitchens*. All the domains contain 2,000 samples for training. Specifically, we train the model by 2,000 labeled source-domain examples and 2,000 unlabeled target-domain examples. We test them on separate target test datasets (4,465 examples in *books*, 3,586 examples in *DVDs*, 5,681 examples in *electronics*, and 5,945 examples in *kitchens*). The data are pre-processed by the method used in [54].

We compare our method with DANN [18] and the Neural Network (NN) [57]. NN is a standard neural network with one hidden layer and does not use any unlabeled target sample for learning, i.e., it is trained by the source-domain data and tested by the target-domain data. We then evaluate DPDA and G-DPDA by performing extensive experiments.

2) Network Architectures: In our experiments, the feature extractor has one hidden layer with 50 neurons. For the domain classifier and label prediction, we stick to one fully connected layer ($x \rightarrow 2$) to do the classification tasks.

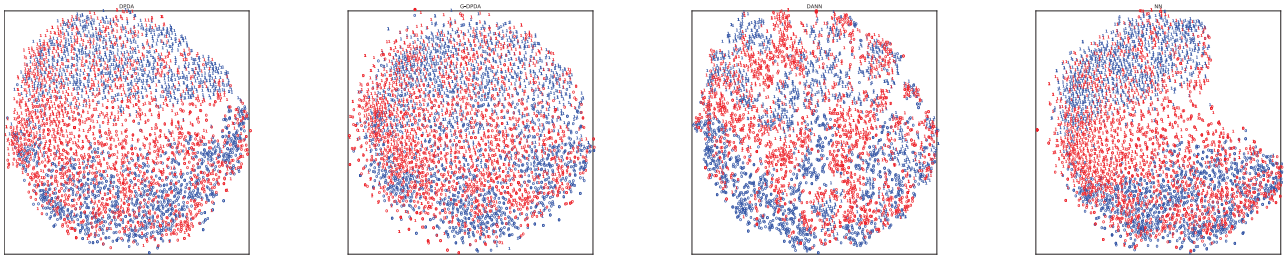
3) Parameter Settings: For both DPDA and G-DPDA algorithms, the adaptation parameter ν is gradually changed from 0 to 1 on a logarithmic scale. As NN is directly trained by source-domain data, it does not have the adaptation parameter. Empirically, the learning rate η is fixed at 0.05 in Phase 1 and 10^{-3} in Phase 2.

In these experiments, we change the value of some parameters as presented in Section V-A.3. To achieve a reasonable trade-off between utility and privacy, we set the clipping parameter as 4 in our model and choose the noise level as 8. We achieve $(8, 10^{-5})$-differential privacy for DPDA, and $(4, 10^{-5})$-differential privacy and $(8, 2 \cdot 10^{-5})$-differential privacy for the target data and the source data of G-DPDA respectively.

4) Visualizations: To demonstrate the transferability of the features learned by DPDA and G-DPDA, we visualize the feature distributions of the fc layer (as shown in Figure 4) and plot the t-SNE [58] embeddings on task $DVD \rightarrow BOOKS$ in Figures 8(a)-(d) for the four methods (DPDA, G-DPDA, DANN, and NN). Feature points of the source and target domains are plotted in blue and red respectively. We can observe that the distance between the two distributions of features obtained by DANN, DPDA, and G-DPDA is much closer than NN.

5) Results and Discussions: Table V shows the target test accuracy of the four methods on the twelve *Amazon review* domain adaptation tasks. We can observe that DPDA has better performance than NN (non-adapted) in most tasks, and it indicates that our DPDA model can complete domain adaptation tasks with a high accuracy while protecting the privacy.

We can observe from Table V that DANN and DPDA have superior performance over NN. The possible reason is that the features from the source and target domains have much closer distributions in DANN and DPDA than NN as illustrated by Figures 8(a), (c) and (d). However, although G-DPDA has a much closer distribution than NN as shown in Figure 8(b), the classification accuracy obtained by G-DPDA is lower than NN as shown in Table V. The reason is that noise is



(a) DPDA (b) G-DPDA (c) DANN (d) NN

Fig. 8. The effect of adaptation reflected by t-SNE visualization. The t-SNE visualizes the distributions of the extracted features of four different methods. Blue points correspond to the source-domain samples, and red points correspond to the target-domain ones.

TABLE V
CLASSIFICATION ACCURACY ON THE AMAZON REVIEWS DATASET

Domain		**Method**			
SOURCE	TARGET	DANN [18]	NN [57]	DPDA (Ours)	G-DPDA (Ours)
BOOKS	DVD	0.809	0.804	0.809	0.769
BOOKS	ELECTRONICS	0.765	0.753	0.758	0.707
BOOKS	KITCHEN	0.779	0.774	0.778	0.720
DVD	BOOKS	0.771	0.732	0.741	0.716
DVD	ELECTRONICS	0.754	0.742	0.749	0.715
DVD	KITCHEN	0.783	0.779	0.783	0.742
ELECTRONICS	BOOKS	0.723	0.724	0.722	0.674
ELECTRONICS	DVD	0.733	0.735	0.728	0.682
ELECTRONICS	KITCHEN	0.858	0.855	0.857	0.818
KITCHEN	BOOKS	0.723	0.717	0.717	0.679
KITCHEN	DVD	0.741	0.739	0.737	0.688
KITCHEN	ELECTRONICS	0.854	0.846	0.846	0.791
Average		0.774	0.767	0.769	0.725

added into both phases in G-DPDA, and the increased noise leads to a decreased performance for the unlabeled target data classification.

The accuracies of DPDA and G-DPDA are lower than DANN due to the added noise, but they have already achieved relatively good results. Intuitively, adding noise into gradients in Phase 2 may have fewer negative effects than Phase 1. This is because the training process in Phase 1 aims to minimize the loss of label prediction (directly related to the label classification performance) while the training process in Phase 2 only aims to decrease the discrimination between the source-domain and target-domain feature distributions.

VI. CONCLUSION

In this paper, a novel differentially private domain adaptation approach called DPDA was proposed to prevent privacy leakages in domain adaptation. We added differentially private noise into gradients in specific layers and training processes in DPDA to protect the private training data, and we used an adversarial-learning strategy to construct domain-invariant features for classifying the unlabeled target data.

We conducted experiments on two popular real-world datasets to evaluate the performance of our proposed method. The results obtained on the *Office-31* dataset demonstrated that DPDA can achieve comparable performance with the state-of-the-art domain adaptation methods while protecting the privacy, with only a modest utility loss. The results obtained on the *Amazon review* datasets showed that both DPDA and G-DPDA have high performance on domain adaptation tasks while protecting the training data privacy.

REFERENCES

[1] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2017, pp. 587–601.

[2] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2015, pp. 1322–1333.

[3] N. Homer *et al.*, "Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays," *PLoS Genet.*, vol. 4, no. 8, 2008, Art. no. e1000167.

[4] S. Sankararaman, G. Obozinski, M. I. Jordan, and E. Halperin, "Genomic privacy and limits of individual detection in a pool," *Nature Genet.*, vol. 41, no. 9, pp. 965–967, Sep. 2009.

[5] Q. Yao *et al.*, "Privacy-preserving stacking with application to cross-organizational diabetes prediction," in *Proc. 28th Int. Joint Conf. Artif. Intell.* San Mateo, CA, USA: Morgan Kaufmann, Aug. 2019, pp. 10–16.

[6] Y. Wang, Q. Gu, and D. Brown, "Differentially private hypothesis transfer learning," in *Proc. ECML PKDD*. Berlin, Germany: Springer, 2018, pp. 811–826.

[7] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. ICML*, 2015, pp. 1180–1189.

[8] I. Mironov, K. Talwar, and L. Zhang, "Rényi differential privacy of the sampled Gaussian mechanism," 2019, *arXiv:1908.10530*. [Online]. Available: http://arxiv.org/abs/1908.10530

[9] T. Tommasi, N. Patricia, B. Caputo, and T. Tuytelaars, "A deeper look at dataset bias," in *Domain Adaptation in Computer Vision Applications*. Berlin, Germany: Springer, 2017, pp. 37–55.

[10] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. ECCV*. Berlin, Germany: Springer, 2014, pp. 818–833.

[11] J. Donahue *et al.*, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. ICML*, 2014, pp. 647–655.

[12] J. Liang, R. He, Z. Sun, and T. Tan, "Aggregating randomized clustering-promoting invariant projections for domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 5, pp. 1027–1042, May 2019.

[13] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 801–814, Apr. 2019.

[14] Z. Ding, N. M. Nasrabadi, and Y. Fu, "Semi-supervised deep domain adaptation via coupled neural networks," *IEEE Trans. Image Process.*, vol. 27, no. 11, pp. 5214–5224, Nov. 2018.

[15] Z. Xu, S. Huang, Y. Zhang, and D. Tao, "Webly-supervised fine-grained visual categorization via deep domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1100–1113, May 2018.

[16] C.-A. Hou, Y.-H.-H. Tsai, Y.-R. Yeh, and Y.-C.-F. Wang, "Unsupervised domain adaptation with label and structural consistency," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5552–5562, Dec. 2016.

[17] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, vol. 1, no. 2, P. 7.

[18] Y. Ganin *et al.*, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2030–2096, May 2015.

[19] B. Banerjee and S. Chaudhuri, "Hierarchical subspace learning based unsupervised domain adaptation for cross-domain classification of remote sensing images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 11, pp. 5099–5109, Nov. 2017.

[20] M. Ghifary, D. Balduzzi, W. B. Kleijn, and M. Zhang, "Scatter component analysis: A unified framework for domain adaptation and domain generalization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 7, pp. 1414–1430, Jul. 2017.

[21] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Scholkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, Jul. 2006.

[22] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," 2014, *arXiv:1412.3474*. [Online]. Available: http://arxiv.org/abs/1412.3474

[23] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. ICML*, 2015, pp. 97–105.

[24] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, "Deep reconstruction-classification networks for unsupervised domain adaptation," in *Proc. ECCV*. Berlin, Germany: Springer, 2016, pp. 597–613.

[25] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Proc. NIPS*. Cambridge, MA, USA: MIT Press, 2018, pp. 1647–1657.

[26] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. NIPS*. Cambridge, MA, USA: MIT Press, 2014, pp. 2672–2680.

[27] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Proc. NIPS*. Cambridge, MA, USA: MIT Press, 2016, pp. 469–477.

[28] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*. Cambridge, MA, USA: MIT Press, 2012, pp. 1097–1105.

[31] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, "DeepCrack: Learning hierarchical convolutional features for crack detection," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1498–1512, Mar. 2019.

[32] Z. Zhang, Q. Zou, Y. Lin, L. Chen, and S. Wang, "Improved deep hashing with soft pairwise similarity for multi-label image retrieval," *IEEE Trans. Multimedia*, vol. 22, no. 2, pp. 540–553, Feb. 2020.

[33] A. A. Cruz-Roa, J. E. A. Ovalle, A. Madabhushi, and F. A. G. Osorio, "A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection," in *Proc. MICCAI*. Berlin, Germany: Springer, 2013, pp. 403–410.

[34] H. Y. Xiong *et al.*, "The human splicing code reveals new insights into the genetic determinants of disease," *Science*, vol. 347, no. 6218, Jan. 2015, Art. no. 1254806.

[35] Q. Zou, L. Ni, Q. Wang, Q. Li, and S. Wang, "Robust gait recognition by integrating inertial and RGBD sensors," *IEEE Trans. Cybern.*, vol. 48, no. 4, pp. 1136–1150, Apr. 2018.

[36] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.

[37] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang, "Membership inference attack against differentially private deep learning model," *Trans. Data Privacy*, vol. 11, no. 1, pp. 61–79, 2018.

[38] Q. Wang, Y. Zhang, X. Lu, Z. Wang, Z. Qin, and K. Ren, "Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, p. 591–606, Jul./Aug. 2018.

[39] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Trans. Inf. Forensics Security*, vol. 15, no. 1, pp. 1486–1500, 2020, doi: 10.1109/TIFS.2019.2939713.

[40] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2016, pp. 308–318.

[41] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," 2016, *arXiv:1610.05755*. [Online]. Available: http://arxiv.org/abs/1610.05755

[42] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011.

[43] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. TCC*. Berlin, Germany: Springer, 2006, pp. 265–284.

[44] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2013.

[45] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Proc. Eurocrypt*. Berlin, Germany: Springer, 2006, pp. 486–503.

[46] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Proc. TCC*. Springer, 2016, pp. 635–658.

[47] T. van Erven and P. Harremos, "Rényi divergence and Kullback–Leibler divergence," *IEEE Trans. Inf. Theory*, vol. 60, no. 7, pp. 3797–3820, Jul. 2014.

[48] I. Mironov, "Rényi differential privacy," in *Proc. IEEE 30th Comput. Secur. Found. Symp. (CSF)*, Aug. 2017, pp. 263–275.

[49] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, p. 116.

[50] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. NIPS*. Cambridge, MA, USA: MIT Press, 2014, pp. 3320–3328.

[51] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. ECCV*. Berlin, Germany: Springer, 2010, pp. 213–226.

[52] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proc. EMNLP*, 2006, pp. 120–128.

[53] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2066–2073.

[54] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," 2012, *arXiv:1206.4683*. [Online]. Available: http://arxiv.org/abs/1206.4683

[55] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," in *Proc. ECCV*. Berlin, Germany: Springer, 2016, pp. 443–450.

[56] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*. [Online]. Available: http://arxiv.org/abs/1207.0580

[57] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 9–48.

[58] L. van der Maaten, "Barnes-hut-SNE," 2013, *arXiv:1301.3342*. [Online]. Available: http://arxiv.org/abs/1301.3342

Qian Wang (Senior Member, IEEE) received the Ph.D. degree from the Illinois Institute of Technology, USA. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. His research interests include AI security, data storage, search and computation outsourcing security and privacy, wireless systems security, big data security and privacy, and applied cryptography. He is a Member of ACM. He received the National Science Fund for Excellent Young Scholars of China in 2018. He is also an expert under National "1000 Young Talents Program" of China. He was a recipient of the 2018 IEEE TCSC Award for Excellence in Scalable Computing (Early Career Researcher), and the 2016 IEEE Asia–Pacific Outstanding Young Researcher Award. He is also a co-recipient of several best paper and best student paper awards from the IEEE ICDCS'17, the IEEE TrustCom'16, WAIM'14, and the IEEE ICNP'11. He serves as an Associate Editors for the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING (TDSC) and the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY (TIFS).

Zixi Li received the B.S. degree from Wuhan University in 2017, where he is currently pursuing the master's degree with the School of Cyber Science and Engineering. His research interests include differential privacy in machine learning, data mining, and privacy preservation.

Qin Zou (Senior Member, IEEE) received the B.E. degree in information engineering and the Ph.D. degree in computer vision from Wuhan University, China, in 2004 and 2012, respectively. From 2010 to 2011, he was a Visiting Ph.D. Student with the Computer Vision Lab, University of South Carolina, USA. He is currently an Associate Professor with the School of Computer Science, Wuhan University. His research activities involve computer vision, pattern recognition, and machine learning. He is a member of the ACM. He is a co-recipient of the National Technology Invention Award of China 2015.

Lingchen Zhao received the B.S. degree from the College of Information Science and Engineering, Central South University, China, in 2016. He is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering, Wuhan University, China. His research interests include differential privacy and its application in data security and applied cryptography.

Song Wang (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign (UIUC) in 2002. From 1998 to 2002, he worked as a Research Assistant with the Image Formation and Processing Group, Beckman Institute, UIUC. In 2002, he joined the Department of Computer Science and Engineering, University of South Carolina, where he is currently a Professor. His research interests include computer vision, medical image processing, and machine learning. He is a Senior Member of the IEEE Computer Society. He is currently serving as an Associate Editor for the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, *Pattern Recognition Letters*, and *Optics Letters*.