Unsupervised Domain Adaptation with Adversarial Residual Transform Networks

Guanyu Cai, Yuqin Wang, Mengchu Zhou, *Fellow, IEEE*, Lianghua He

Abstract—**Domain adaptation is widely used in learning problems lacking labels. Recent researches show that deep adversarial domain adaptation models can make markable improvements in performance, which include symmetric and asymmetric architectures. However, the former has poor generalization ability whereas the latter is very hard to train. In this paper, we propose a novel adversarial domain adaptation method named Adversarial Residual Transform Networks (ARTNs) to improve the generalization ability, which directly transforms the source features into the space of target features. In this model, residual connections are used to share features and adversarial loss is reconstructed, thus making the model more generalized and easier to train. Moreover, regularization is added to the loss function to alleviate a vanishing gradient problem, which enables the training process stable. A series of experimental results based on Amazon review dataset, digits datasets and Office-31 image datasets show that the proposed ARTN method greatly outperform the methods of the state-of-the-art.**

Index Terms—**Adversarial neural networks, unsupervised domain adaptation, residual connections, transfer learning.**

I. Introduction

ALTHOUGH deep neural networks trained on large-scale labeled datasets could achieve excellent performance across varieties of tasks, such as sentiment analysis [1], [2], image classification [3], [4] and semantic segmentation [5]. They usually fail to generalize well on novel tasks according to [6]. A convincing explanation is that there exists domain shift between training data and testing one [7], [8]. To alleviate the negative effect raised by domain shift, domain adaptation (DA) is proposed to utilize labeled data from the source domain to make models generalize well on target domain [9].

Domain adaptation, which is a field belongs to transfer learning, has long been utilized to make it possible to exploit the knowledge learned in one specific domain to effectively improve the performance in a related but different domain. Earlier methods of DA aim to learn domain-invariant feature representations from data by jointly minimizing a distance metric that actually measures the adaptability between a pair of source and target domains, such as Transfer Component Analysis [10], Geodesic Flow Kernel [11], and Transfer Kernel Learning [12]. In order to learn transferable features well, researchers apply deep neural networks to DA models [13]–[15]. A feature extractor neural network is trained by reducing "distance" between distributions of two different domains, on the assumption that the classifier trained by source data also works well in the target domain. In this kind of methods, Maximum Mean Discrepancy (MMD) loss is widely used for mapping different distributions [16]. For example, Deep Adaptation Networks (DAN) [17], Joint Adaptation Networks [18] and Residual Transfer Networks [19] apply MMD loss to several layers whereas Large Scale Detection through Adaptation [20] adds a domain adaptation layer that is updated based on MMD loss.

Recently, the idea of Generative Adversarial Networks (GANs) [21], [22] has been widely applied to DA. Methods of using GANs to transform source images to target ones are proposed [23], [24], whose classifiers are trained with the generated target images. However, when distributions of the source and target domains are totally different, adversarial training easily gets stuck because of a gradient vanishing phenomenon. Alternative methods train GANs on features of the source and target domains, whose generator is acted as a feature extractor, and discriminator is acted as a domain classifier. There are two kinds of architectures in adversarial domain adaptation, which can effectively adapt source and target distributions. One is symmetric adaptation where features in the source and target domains are generated from the same network [25], [26], and the other is asymmetric adaptation where features are generated from different networks [27]. However, the former is poor at generalization whereas the latter is difficult to train.

To solve the above problems, in this work, we propose a novel feature-shared model for adversarial domain adaptation, which achieves the flexibility of asymmetric architecture and can be easily trained. In the proposed framework as shown in Fig. 1, a weight-shared feature extractor distills features from different domains, and a feature-shared transform network maps features from the source domain to the space of target features. Adversarial learning is completed with the losses from the label and domain classifiers. Note that we design residual connections between the feature extractor and transform network to ease the learning of distribution mapping by sharing features. In addition, in order to avoid getting stuck in local minima, we construct a regularization term to ensure that the model at least knows a vague, if not exact, direction to match different distributions and avoid gradient vanishing. The main contributions of this work are as follows:

1) A novel adversarial model that learns a non-linear mapping from a source domain to a target one is proposed. By using the features generated from the source domain, this model is more generalized in the target domain.
2) During training, concise regularization that ensures the model to select the shortest path from all the transfer paths is constructed, helping to stabilize the adversarial optimization.
3) We extensively evaluate the proposed method on several standard benchmarks. The results demonstrate that

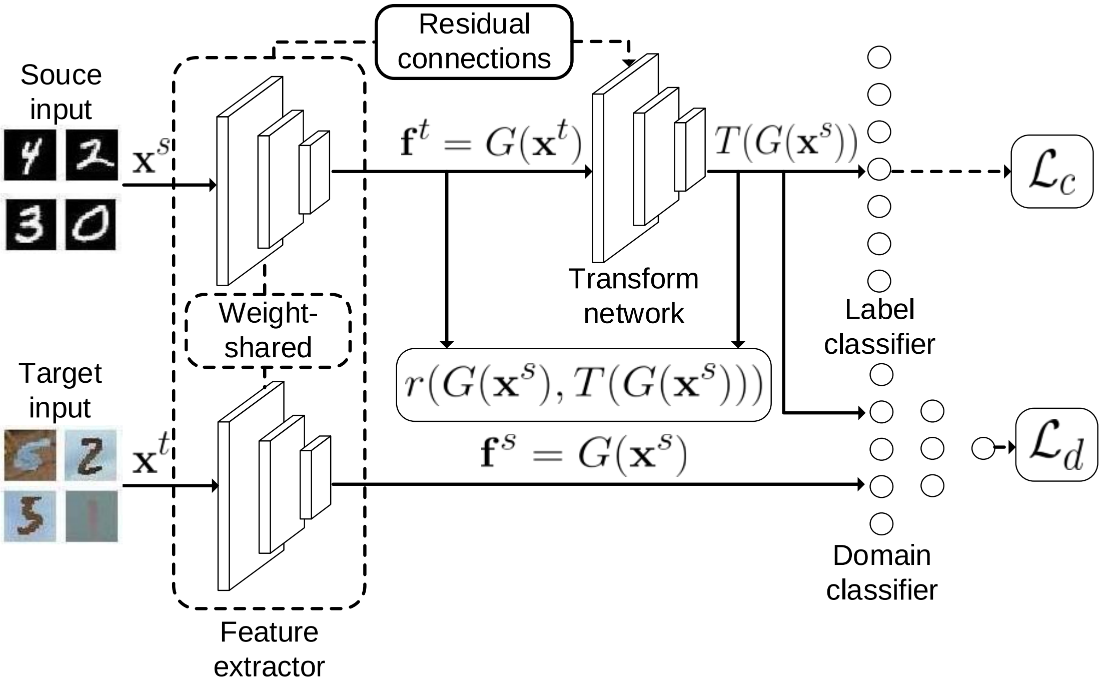


Fig. 1. The architecture of the proposed model. First feature extractor G distills the feature representations of source and target images. Then, transform network T projects source features $\mathbf{f}^s = G(\mathbf{x}^s)$ to the space of target features $\mathbf{f}^t = G(\mathbf{x}^t)$. Finally, the label classifier C is trained with the fake target features $T(G(\mathbf{x}^s))$ and predicts the labels of target features $G(\mathbf{x}^t)$ during the test period. In addition, domain classifier D is trained to distinguish fake target features $T(G(\mathbf{x}^s))$ and real target features $G(\mathbf{x}^t)$, which can minimize the discrepancy between source and target domains through adversarial optimization. The regularization term $r(G(\mathbf{x}^s), T(G(\mathbf{x}^s)))$ measures the distance between $\mathbf{f}^s = G(\mathbf{x}^s)$ and $T(G(\mathbf{x}^s))$.

our model outperforms the state-of-the-art methods in accuracy. Notably, our model can maintain excellent generalization and anti-noise abilities.

The rest of this paper is organized as follows. Section II reviews some related work on unsupervised domain adaptation. In Section III, we detail the proposed adversarial residual transform networks. The training strategy is also presented in this section. Section IV demonstrates experimental results in several standard benchmarks as well as comparisons against state-of-the-art methods. At last, we conclude this paper in Section V.

II. Related Work

DA has been extensively studied in recent years. Several studies give theoretical analyses of the error bound when the training and testing data are drawn from different distributions [7], [8]. This means that it is feasible to utilize the knowledge across domains. Moreover, the most important problem in DA is how to reduce the discrepancy between the source and target domains. Therefore, many methods modify a classifier to match different distributions [28]–[31]. Recently, because deep neural networks can learn feature representations including information identification, and these features are also transferable [6], they are widely used in DA methods. These methods have received much research on how to measure the distance between different domains and how to design a network structure. A number of methods have indeed achieved remarkable performance improvement.

A. Traditional Domain Adaptation

A natural idea is to reweight instances in the source domain, where instances similar to the target distribution are considered more important. This kind of method has proven effective for adaptation for the differences between the source and target distributions. Some methods reweight instances by direct importance weighting algorithm, such as [32], [33], [34] and [30]. There are also some other methods reweighting importance to adapt the loss function with noisy labels [35], or to reduce the target and conditional shift [36].

Another idea to domain adaptation is to explicitly make the source and target distributions similar. Many statistic characteristics are chosen to be the metrics to align subspaces of different distributions. MMD, which measures the difference of expectations in reproducing kernel Hilbert space (RKHS) of the source and target distributions, is widely used in many methods [10], [12], [28], [31]. In order to align more complex statistic characteristics, [37], [38], and [42] employ moments of varying order to match different distributions, which are easy to implement with low computational complexity. Instead of exploiting statistic characteristics, some methods [11], [39], [40] utilize manifold learning to transform the source distribution similar to target distribution. In these methods, feature spaces are refined into low-dimensional spaces and feature distortion is avoided.

B. Deep Domain Adaptation

Recently, development of deep neural networks promotes deep domain adaptation. In [41], the experiment results demonstrate that using features of deep neural networks instead of hand-crafted ones alleviates the negative influences of domain shift even without any adaptation. Besides relying on pre-trained deep features, a number of methods combine statistic characteristics and deep neural networks to a unified framework, which greatly improve the performance on varying tasks. In [17]–[20], MMD is embedded in deep convolutional neural networks. In [42] and [43], high order moments are utilized to align feature spaces of the source and target domains. In addition, some methods design special architectures to minimize the discrepancy between the source and target domains. In [44], a Siamese architecture is introduced to adapt pairs of source and target instances. In [45], [46], autoencoders are suggested to learn the transformation from the source to target domain.

Some methods choose adversarial loss to learn manifest invariant factors underlying different populations, which usually add a subnetwork acting as a domain discriminator. In these models, deep features are learned to confuse the domain discriminator such that they could capture the most discriminative information about classification instead of characteristics of domains. Domain-Adversarial Neural Networks (DANN) [26] consist of a symmetric feature generator, label discriminator, and domain discriminator. The whole model can be directly optimized via a gradient reversal algorithm. Deep Reconstruction-Classification Networks [47] also take adversarial learning but add a reconstruction step for target images. Adversarial Discriminative Domain Adaptation (ADDA) [27] uses an asymmetric feature generator that is trained alternatively with the domain classifier. The above domain adversarial networks fall into two categories. Some adversarial adaptation methods [25], [26] share weights between source and target feature extractors. They use the same network to learn representations from different inputs, which learns a symmetric transformation to utilize the transferability of features generated from deep neural networks and reduces parameters in the model. The other methods construct two networks for source and target domains, respectively [23], [24], [27]. They can learn an asymmetric transformation, allowing networks to learn parameters for each domain individually. In theory, asymmetric transformation can lead to more effective adaptation [48].

Adversarial domain adaptation has also been explored in generative adversarial networks (GANs). Coupled Generative Adversarial Networks (CoGANs) [23] apply GANs to DA which train two GANs to generate the source and target images, respectively. Pixel-Level Domain Adaptation [24] uses a conditional GAN model to synthesize target images to facilitate training a label classifier. Methods based on GANs can improve the performance of digits datasets, but their downside is a difficult training process as caused by gradient vanishing when facing more natural image datasets [49]. In this work, we focus on learning the mapping of different feature spaces instead of synthesizing target images, and propose a discriminative model aiming to adapt distinct domains.

III. Adversarial Residual Transform Networks

In this section, we describe the details of our model. We first introduce the definitions of unsupervised domain adaptation and the preliminary domain adversarial networks, and then demonstrate the key innovations of our model, which can well handle the problems of previous models. At last, we give a complete algorithm of matching the distributions of target and source domains using the proposed model.

A. Definitions

When it comes to a machine learning task, a domain D corresponds to four parts: feature space $\mathcal{X}$, label space $\mathcal{Y}$, marginal probability distribution $P(\mathbf{X})$ and conditional probability distribution $P(\mathbf{X}|\mathbf{Y})$, where $\mathbf{X} \in \mathcal{X}$, $\mathbf{Y} \in \mathcal{Y}$. We always use subscript s and t to denote the source and target distributions. In traditional machine learning task, training data are drawn from the source domain D_s and testing data are drawn from the target domain D_t, where their marginal and conditional probability distributions are the same ($P_s(\mathbf{X}^s) = P_t(\mathbf{X}^t), P_s(\mathbf{X}^s|\mathbf{Y}^s) = P_t(\mathbf{X}^t|\mathbf{Y}^t)$). Thus, models trained in the source domain are feasible to the target domain. However, in unsupervised DA, these assumptions are not valid, which leads us to a more difficult problem as follows.

Given a source domain as $D_s\{\mathbf{x}_i^s, \mathbf{y}_i^s\}_{i=1}^{n_s}$, where n_s is the number of source domain samples, $\mathbf{x}_i^s$ is the ith instance in the source domain and $\mathbf{y}_i^s$ is the label of $\mathbf{x}_i^s$. Similarly, a target domain is denoted as $D_t\{\mathbf{x}_i^t\}_{i=1}^{n_t}$, where n_t is the number of target domain samples, $\mathbf{x}_i^t$ is the ith instance in the target domain. The source and target domains are drawn from distribution $P_s(\mathbf{X}^s)$ and $P_t(\mathbf{X}^t)$, respectively, which are different. In most cases, conditional probability distributions are also different ($P_s(\mathbf{X}^s|\mathbf{Y}^s) \neq P_t(\mathbf{X}^t|\mathbf{Y}^t)$) The goal is to learn a feature extractor G_t and a classifier C_t for the target domain. The feature extractor G_t distills feature representations $\mathbf{f}^t = G_t(\mathbf{x}^t)$ from target images, and the classifier C_t correctly predict the labels of target images receiving $\mathbf{f}^t = G_t(\mathbf{x}^t)$. Because of the lack of annotations in the target domain, DA learns a feature extractor G_s and a classifier C_s with samples from the source domain, and tries to adapt them to be useful in the target domain.

B. Adversarial Domain Adaptation

To solve unsupervised DA problem, a bunch of methods are proposed. One of the most effective methods is adversarial domain adaptation, and we modify this kind of framework in this paper to improve its generalization and anti-noise ability. In domain adversarial models, in order to make C_s valid in a target domain, these models usually train a feature extractor G, a label classifier C and a domain classifier D. In details, these models set $G = G_t = G_s$, and $C = C_t = C_s$, which means the feature extractor and label classifier are used for both source and target domains. Specifically, D also receives feature representations from G and is trained to minimize the discrepancy between source and target feature distributions:

$G(\mathbf{x}^s)$ and $G(\mathbf{x}^t)$. An adversarial training procedure is a minimax two-player game [21], where D is trained to discriminate samples from different domains, and G is trained to fool D. To train the whole network in an end-to-end way, DANN [26] constructs the following loss function:

$$\begin{aligned}\mathcal{L}(\theta_d, \theta_g, \theta_c) =& \frac{1}{n_s} \sum_{\mathbf{x}_i \in D_s} \mathcal{L}_c(C(G(\mathbf{x}_i)), y_i) - \\ & \frac{\lambda}{n} \sum_{\mathbf{x}_i \in D_s \cup D_t} \mathcal{L}_d(D(G(\mathbf{x}_i)), d_i) \end{aligned} \tag{1}$$

where $n = n_s + n_t$, and λ is a trade-off parameter between label loss $\mathcal{L}_c$ and domain loss $\mathcal{L}_d$. θ_d, θ_g and θ_c are the parameters of D, G and C, respectively. y_i and d_i denote the class and domain labels of images. After convergence, optimal parameters $\hat{\theta}_d$, $\hat{\theta}_c$ and $\hat{\theta}_g$ can deliver a saddle point given as:

$$\hat{\theta}_d = \arg\min_{\theta_d} \mathcal{L}_d(\theta_d, \theta_g) \tag{2}$$

$$\hat{\theta}_c = \arg\min_{\theta_c} \mathcal{L}_c(\theta_g, \theta_c) \tag{3}$$

$$\hat{\theta}_g = \arg\min_{\theta_g} \mathcal{L}(\theta_d, \theta_g, \theta_c) \tag{4}$$

In such framework, a DA model can be trained in an end-to-end way. Several models based on this kind of architecture have achieved top performances in different visual tasks [26], [47].

C. Residual Connections

In our model, we do not just rely on feature extractor G to map different distributions. Instead, we construct an adversarial residual transform network (ARTN) T to project source features $\mathbf{f}^s = G(\mathbf{x}^s)$ to the space of target features. The network is trained to generate fake target features $T(G(\mathbf{x}^s))$, which are in the same distribution as real target features $\mathbf{f}^t = G(\mathbf{x}^t)$. Then, we use the fake target features $T(G(\mathbf{x}^s))$ and corresponding labels $\mathbf{y}^s$ to train a classifier C for the target domain. After training, the labels of target samples are predicted by C.

In previous unsupervised DA methods, the weights of feature extractor G for the source and target domains are shared [17]–[20]. In detail, they train just one neural network to distill representations for different domains. In such symmetric architecture, models are stable during the training period and seldom diverge. However, regarding matching different distributions, the generalization ability of asymmetric transformation is better than that of symmetric one [27]. This means that constructing two networks for source and target domains instead of an identical network, is possible to enhance the generalization ability. In real applications, the training and testing data are possible to have a large discrepancy, which suggests that our model should be more general. Therefore, in order to do so, our network is designed to have an asymmetric architecture.

However, in deep neural networks, feature representations generated from layers are transferable, which means that there exists domain-invariant information in features [6]. If the networks are trained to capture domain-invariant information from the source features and utilize them to classify the target samples, there would be a boost to generalization. In addition, even though the training does not converge in asymmetric models, the shared source features retain useful information for classifying and preventing models from being collapsed. However, the asymmetric architecture proposed in [27] is hard to obtain such enhancement and the feature extractor for target domain is easy to collapse, because there exists no relationship between the feature extractors of the source and target domains. In order to make our asymmetric model learn domain-invariant information and avoid diverging during training, we propose a transform network which builds connections between source and target domain features.

As for delivering features between networks, residual connections proposed by [3] are proved effective. They concatenate features from one network to another to share information. In such way, original features from feature extractor are conveyed to transform network. At the same time, the shortcut connection with no parameters helps gradients flow. Thus, by utilizing residual connections, the proposed transform network is able to share features between two different networks, which helps capture domain-invariant information. In summary, the proposed model builds an asymmetric transformation that consists of a feature extractor and a transform network. Residual connections between the feature extractor and transform network share features for capturing domain-invariant information.

The detailed architecture of residual connections between the feature extractor and transform network is shown in Fig. 2. The weight-tied feature extractor G is trained to capture representations from source samples $\mathbf{x}^s$ and target samples $\mathbf{x}^t$. The transform network stacks a few layers by using the same architecture with the feature extractor. Unlike the symmetric transformation, the proposed transform network shares features with the feature extractor instead of parameters. Our transform network is also different from asymmetric transformation where two networks have no relationship. We add residual connections between the feature extractor and transform network to share features. Therefore, with carefully designed architecture, our model is able to simultaneously alleviate drawbacks of symmetric and asymmetric models.

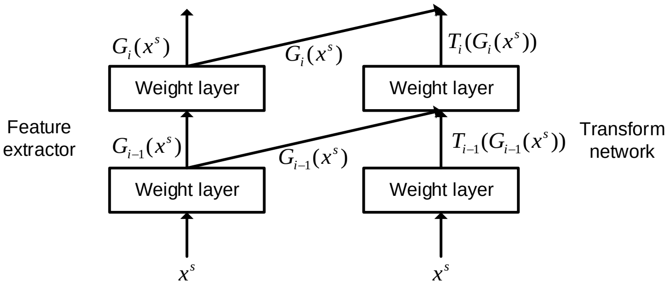


Fig. 2. Residual connections between the feature extractor and transform network.

Theoretically, by denoting the desired underlying mapping between source and target distributions as M and letting $G(\mathbf{x}^t) = M(G(\mathbf{x}^s))$, we intend to train the transform network T to fit a mapping of $T(G(\mathbf{x}^s)) = M(G(\mathbf{x}^s)) - G(\mathbf{x}^s)$. In details, for an N layer transform network, the ith layer of a transform network where $i < N$ is defined as:

$$T_i(G_i(\mathbf{x}^s)) = T_i(T_{i-1}(G_{i-1}(\mathbf{x}^s))) + G_i(\mathbf{x}^s) \tag{5}$$

where $T_i(\cdot)$ and $G_i(\cdot)$ denote the ith layer of the transform network and feature extractor individually. The inputs of T are original data, and the output is $T_N(G_N(\mathbf{x}^s))$. In detail, for both the fully connected neural networks and convolutional neural networks, outputs of each layer from feature extractor are element-wise added to the corresponding layer of transform network.

D. Adversarial Losses

Once we have parametrized G and T, we employ adversarial loss to adapt different distributions. The architecture modification requires us to revise our loss function. Instead of measuring the distance between source features $\mathbf{f}^s = G(\mathbf{x}^s)$ and target features $\mathbf{f}^t = G(\mathbf{x}^t)$ generated from one feature extractor, the proposed model lets domain classifier D discriminate source features $\mathbf{f}^s = T(G(\mathbf{x}^s))$ from the transform network and target features $\mathbf{f}^t = G(\mathbf{x}^t)$ from the feature extractor. Therefore, the loss function is modified from (1):

$$\begin{aligned}\mathcal{L}(\theta_d, \theta_g, \theta_c) = & \frac{1}{n_s} \sum_{\mathbf{x}_i \in D_s} \mathcal{L}_c(C(T(G(\mathbf{x}_i))), y_i) - \\ & \frac{\lambda}{n} (\sum_{\mathbf{x}_i \in D_s} \mathcal{L}_s(D(T(G(\mathbf{x}_i))), d_i^s) + \\ & \sum_{\mathbf{x}_i \in D_t} \mathcal{L}_t(D(G(\mathbf{x}_i))), d_i^t))\end{aligned} \tag{6}$$

where d_i^s and d_i^t denote the domain labels of the ith source and target samples, respectively. $\mathcal{L}_s$ and $\mathcal{L}_t$ denote the domain loss of source and target samples, respectively. This objective funtion replaces $G(\mathbf{x}_i)$ in (1) with $T(G(\mathbf{x}_i))$, indicating that our model uses features generated from transform network T to be the input of label classifier C and domain classifier D.

When (6) is used as the loss function for adversarial training, similar to GANs, the adversarial training of unsupervised DA can cause a gradient vanishing issue [49]. When the distributions of source and target domains are totally different, the domain classifier can easily distinguish the samples from the source and target domains such that $\mathcal{L}_s$ and $\mathcal{L}_t$ would be very small, thereby causing the gradients to vanish.

To address this issue, we add a regularization term to the loss function based on the optimal transport problem as defined by Monge. DA's goal is to find a mapping from a source domain to a target domain, while the optimal transport problem gives a solution that transfers one distribution to another. Therefore, that problem can be represented in the form of Monge's formulation of the optimal transport problem [39], [40]. If we denote the probability measures over P_s and P_t as μ_s and μ_t, respectively, Monge's formulation of the optimal transport problem is [39]:

$$T_0 = \arg\min_T \int_{\mathbf{X_s}} r(\mathbf{x}, T(\mathbf{x})) d\mu(\mathbf{x}), s.t. T(\mu_s) = \mu_t \tag{7}$$

where $r(\cdot)$ denotes some kind of distance metric, T denotes a transport mapping from P_s to P_t, and T_0 is the optimal solution of T. $\mathbf{x}$ denotes the samples drawn from P_s. DA's goal is to find a transport mapping T_0 satisfying $T(\mu_s) = \mu_t$, which means that a transformation from source distribution P_s to target distribution P_t should be found. Specifically, in our model, we use transform network T to fit the transport mapping to meet $T(\mu_s) = \mu_t$ via adversarial training. By fitting $r(\cdot)$, we can construct a regularization term that measures the distance between $G(\mathbf{x}^s)$ and $T(G(\mathbf{x}^s))$. In our model, $r(\cdot)$ is the cosine distance between them:

$$r(G(\mathbf{x}^s), T(G(\mathbf{x}^s))) = -\frac{\langle G(\mathbf{x}^s \cdot T(G(\mathbf{x}^s)) \rangle}{|G(\mathbf{x}^s)| \cdot |T(G(\mathbf{x}^s))|} \tag{8}$$

where $\langle \cdot \rangle$ denotes inner product, and $|\cdot|$ denotes L_2 norm.

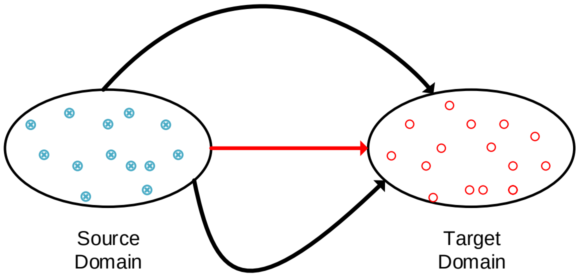


Fig. 3. When transferring features from the source to target domain, the regularization term proposed forces our model choosing the shortest path (red line).

For a transport problem, there are usually a few practical paths as shown in Fig. 3. The optimal transport theory seeks the most efficient way of transforming one distribution of mass to another, just like the red line in Fig. 3. In detail, $\int r(\mathbf{x}, T(\mathbf{x})) d\mu(\mathbf{x})$ in the Monge formulation (7) which indicates the expected cost of transportation is to be minimized. If $\int r(\mathbf{x}, T(\mathbf{x})) d\mu(\mathbf{x})$ reaches the minimum, the most efficient path would be found. Once we refine the optimal transport theory into unsupervised DA, the regularization term $r(G(\mathbf{x}^s), T(G(\mathbf{x}^s)))$ leads our model to select the most efficient way of transforming the source to target distribution. Specifically, the regularization term attempts to constrain the distance between the features before and after the transformation. If we regard the distance as the cost of the transformation, similar to the cost of transportation, the regularization term attempts to select the shortest path from a number of transfer paths which map a source to target distribution.

On one hand, when the distributions of source and target domains are totally different, domain classifier D can so easily distinguish samples from different domains that $\mathcal{L}_s$ and $\mathcal{L}_t$ backpropagate very small gradients. In this situation, the regularization term $r(G(\mathbf{x}^s), T(G(\mathbf{x}^s)))$ can still provide gradients to the target mapping. On the other hand, when D directs updating parameters, the regularization term would constrain the range of updating to prevent features from changing too rapidly, because it constrains differences between the features before and after the transformation. Thus, the stability of the training procedure of the proposed model is guaranteed via such added regularization term.

Therefore, our objective function is

$$\begin{aligned}\mathcal{L}(\theta_d, \theta_g, \theta_c) =& \frac{1}{n_s} \sum_{\mathbf{x}_i \in D_s} \mathcal{L}_c(C(T(G(\mathbf{x}_i))), y_i) - \\ & \frac{\lambda}{n} (\sum_{\mathbf{x}_i \in D_s} \mathcal{L}_s(D(G(\mathbf{x}_i)), d_i^s) + \\ & \sum_{\mathbf{x}_i \in D_t} \mathcal{L}_t(D(T(G(\mathbf{x}_i))), d_i^t)) + \\ & \beta \cdot r(G(\mathbf{x}^s), T(G(\mathbf{x}^s)))\end{aligned} \tag{9}$$

where β denotes the coefficient parameter of a regularization term. In addition, the optimization problem is to find parameters $\hat{\theta}_g$, $\hat{\theta}_d$ and $\hat{\theta}_c$, where $\hat{\theta}_c$, $\hat{\theta}_g$, and $\hat{\theta}_d$ satisfy:

$$\hat{\theta}_d = \arg\min_{\theta_d} (\mathcal{L}_s(\theta_d, \theta_g) + \mathcal{L}_t(\theta_d, \theta_g)) \tag{10}$$

$$\hat{\theta}_c = \arg\min_{\theta_c} \mathcal{L}_c(\theta_g, \theta_c) \tag{11}$$

$$\hat{\theta}_g = \arg\min_{\theta_g} \mathcal{L}(\theta_d, \theta_g, \theta_c) \tag{12}$$

E. Complete Algorithm

In training period, our model comes with two parts. In the first stage, the feature extractor G and transform network T receives labeled source samples from $D_s\{\mathbf{x}_i^s, y_i^s, d_i^s\}_{i=1}^{n_s}$, and outputs $\mathbf{f}^s$ and $T(\mathbf{f}^s)$. With class labels y^s and domain labels d^s, $\mathcal{L}_c$ is computed by label classifier C, and $\mathcal{L}_s$ is computed by domain classifier D. At the same time, the regularization term $r(G(\mathbf{x}^s), T(G(\mathbf{x}^s)))$ is also obtained according to $\mathbf{f}^s$ and $T(\mathbf{f}^s)$. In the second stage, the feature extractor G receives unlabeled samples from $D_t\{\mathbf{x}_i^t, d_i^s\}_{i=1}^{n_t}$, and outputs $\mathbf{f}^t$. Similarly, $\mathcal{L}_t$ is computed by domain classifier D. At last, all the above losses are multiplied by their corresponding coefficients, and then the model is optimized using these losses.

As for optimizing adversarial networks, previous studies have carried out a number of exploration [26], [27]. In [27], an iterative optimization strategy is proposed, where the feature extractor and domain classifier updates their parameters iteratively. This is the most common training strategy which also widely used in GANs [21], [22]. One of the obstacles to this strategy is that tuning the interval steps between the domain classifier updates parameters. Unsuitable interval steps may cause failure of model training so that we have to tune this hyperparameter for each model carefully. Instead, in [26], the proposed gradient reversal layer (GRL) replaces iterative optimization. During the forward propagation, GRL has no different with normal layers, whereas during the backpropagation, GRL reverses the gradient from the subsequent layer, multiplies it by a coefficient γ and passes it to the previous layer. Based on a large number of experiments, [26] adjusts γ using the following formula: $\gamma = \frac{2}{1+e^{-10p}} - 1$, where p is the training progress linearly changing from 0 to 1. In the implementation of ARTN, we choose GRL to optimize our model. With this strategy, there is no need to tune the interval steps, and parameters of the feature extractor and domain classifier are updated in once backpropagation.

Algorithm 1 provides the pseudo-code of our proposed learning procedure. With stochastic gradient descent (SGD), parameters θ_d, θ_c and θ_g are updated. When the loss converges, the training stops.

Algorithm 1 Learning Procedure of ARTN

```
Input:
    Labeled source samples $D_s\{\mathbf{x}_i^s, y_i^s, d_i^s\}_{i=1}^{n_s}$
    Unlabeled target samples $D_t\{\mathbf{x}_i^t, d_i^s\}_{i=1}^{n_t}$
    Learning rate $\alpha$, Coefficient parameters $\lambda, \beta$
Output:
    Model parameters $\{\theta_d, \theta_g, \theta_c\}$
while not converge do
    for $i$ from 1 to $n_s$ do
        $\mathbf{f}_i^s = G(\mathbf{x}_i^s)$
        $\mathcal{L}_c = crossentropy(C(T(\mathbf{f}_i^s)), y_i^s)$
        $\mathcal{L}_s = crossentropy(D(T(\mathbf{f}_i^s)), d_i^s)$
        $reg = r(\mathbf{f}_i^s, T(\mathbf{f}_i^s))$
    end for
    for $i$ from 1 to $n_t$ do
        $\mathbf{f}_i^t = G(\mathbf{x}_i^t)$
        $\mathcal{L}_t = crossentropy(D(\mathbf{f}_i^t), d_i^t)$
    end for
    $\mathcal{L}_d \leftarrow \mathcal{L}_s + \mathcal{L}_t$
    $\theta_d \leftarrow \theta_d - \alpha \cdot \frac{\partial \mathcal{L}_d}{\partial \theta_d}$
    $\theta_c \leftarrow \theta_c - \alpha \cdot \frac{\partial \mathcal{L}_c}{\partial \theta_c}$
    $\theta_g \leftarrow \theta_g - \alpha \cdot \frac{\partial (\mathcal{L}_c - \lambda \mathcal{L}_d + \beta reg)}{\partial \theta_g}$
end while
```

IV. Experiments

In order to evaluate the effectiveness of the proposed method, we test the Adversarial Residual Transform Network (ARTN) for unsupervised DA in several difficulty experiments. For the first experiment, we test our model in a sentiment analysis task. Second, to test the performance of ARTN when the source and target domains are relatively similar, the model is evaluated on several digits datasets. Third, to test it when facing a large discrepancy between the source and target domains, the model is evaluated on a natural image dataset. Fourth, to test the anti-noise and generalization abilities, we test it when target images are added with varying noise. Finally, to test the effectiveness of regularization in the proposed method, we compare the performance of ARTN with and without regularization on a natural image dataset. In all experiments, we implement models with Pytorch, and employ the learning strategy GRL mentioned in Section III, which reverses and propagates gradients to the feature extractors.

A. Sentiment Analysis Experiment

We use the **Amazon reviews** dataset with the same preprocessing used in mSDA [50] and DANN [26]. **Amazon reviews** contains reviews of four different categories of products: `books`, `DVDs`, `kitchen appliances` and `electronics`, which means that this dataset includes four domains and we can set up twelve domain adaptation tasks across these domains. Reviews are encoded in 5 000 dimensional feature vectors of unigrams and bigrams, and labels are binary: 0 if the product is ranked up to 3 stars, and 1 if the product is ranked 4 or 5 stars. In all twelve tasks, we use 2000 labeled source samples and 2000 unlabeled target samples to train our model. In testing period, we test our model on separate target test sets (between 3000 and 6000 examples). To evaluate the effectiveness of our model, we compare it with DANN [26] and the model with no adaptation. The results are directly cited from the original pulication.

In this experiment, we use the same neural network with DANN [26]. Both domain and label classifiers consist of just one layer with 100 hidden units followed by the final output layer. Because there is only one hidden layer in the neural network, we build just one residual connection. ReLU activation function and batch normalization are employed. We choose SGD with 0.001 learning rate as the optimizer, and the momentum is set to 0.9. The coefficient parameter λ is set to 0.5, and β is set to 0.1. The batch size is set to 128. All the results are recorded after convergence.

Results of the sentiment analysis experiment are shown in Table I. Our accuracy of the ARTN model is the highest in ten out of twelve domain adaptation tasks. In the remaining two tasks, our model achieves tie of highest accuracy in `kitchen appliances→DVDs`, and achieves second highest accuracy in `electronics→DVDs`.

B. Digits Experiment

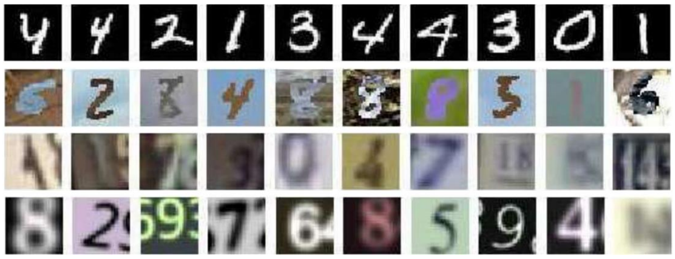


Fig. 4. Samples of digits dataset. The first to last rows correspond to MNIST, MNIST-M, SVHN and SYN NUMS.

In order to evaluate the performance when the discrepancy between source and target domains is relatively small, we experimentally test our proposed method in several pairs of unsupervised domain adaptation tasks whose images are from the **MNIST**, **MNIST-M**, **SVHN** and **SYN NUMS** digits datasets. All these datasets consist of 10 classes, and we use the full training sets in all datasets. Example images from each dataset are shown in Fig. 4. In this experiment, we set three transfer tasks: MNIST→MNIST-M, SVHN→MNIST, and SYN NUMS→SVHN. As is shown in Fig. 4, images in SYN NUMS and SVHN are similar, whereas images in MNIST are much different from the other digits datasets.

We choose several unsupervised DA approaches to compare with the proposed one. CORAL [42] and DAN [17] rely on the distance metric between source and target distributions. DANN [26], CoGAN [23] and ADDA [27] are based on adversarial learning. The results are cited from each study.

For MNIST→MNIST-M, we use a simple modified LeNet [51]. As for the domain classifier, we stack two fully connected layers: one layer with 100 hidden units followed by the final output layer. Each hidden unit uses a ReLU activation function. For SVHN→MNIST and SYN NUMS→SVHN, we use a three-layer convolutional network as a feature extractor, and a three-layer fully connected network as a domain classifier. In all tasks, batch normalization is employed. We employ SGD with 0.01 learning rate and the momentum is set to 0.9. The coefficient parameter λ is set to 2, and β is set to 0.5. The batch size is set to 128. Metric of the experiment is prediction accuracy in the target domain, which is reported after convergence.

Results of our digits experiment are shown in Table II. In SVHN→MNIST and MNIST→MNIST-M, the proposed model's accuracy is 81.4% and 79.4%, respectively, which outperforms the best of other methods by 4.7% and 3.4%. In SYN NUMS→SVHN, its accuracy achieves 89.1%, which is comparable to DANN's. Thus, in two of three tasks, our approach outperforms other methods, and in the task whose source and target datasets are similar, it can achieve competitive results.

C. Image Classification Experiment

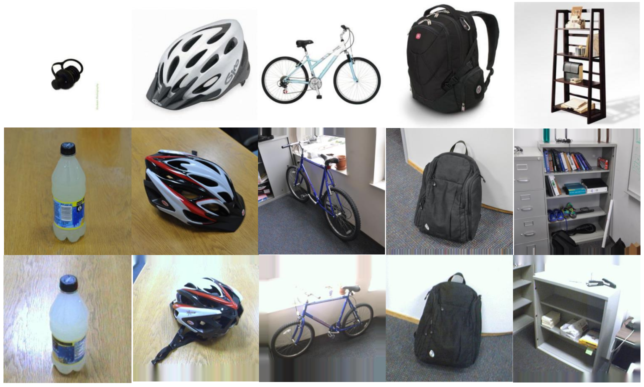


Fig. 5. Samples of Office-31 dataset. The first to last rows correspond to AMAZON, DSLR and WEBCAM.

We further evaluate our model in a more complex setting. The proposed model is tested on a natural image dataset **Office-31**, which is a standard benchmark for visual domain adaptation, comprising 4,110 images and 31 categories collected from three domains: `AMAZON` (**A**, images downloaded from amazon.com) with 2,817 images, `DSLR` (**D**, high-resolution images captured by a digital SLR camera) with 498 images and `WEBCAM` (**W**, low-resolution images captured by a Web camera) with 795 images. Samples of **Offfice-31** dataset are shown in Fig. 5. In order to test the generalization ability of different methods, we focus on the most difficult four tasks [17]: **A**→**D**, **A**→**W**, **D**→**A** and **W**→**A**. In **A**→**W** and **A**→**D**, models are easier to train because images in source domain **A** are adequate. In **W**→**A** and **D**→**A**, there are only hundreds images in the source domain but about 2,900 images in the target domain, such that models are

TABLE I

Classification accuracy percentage of sentiment analysis experiment among all twelve tasks. The first column corresponds to the performance if no adaption is implemented. The proposed method outperforms the others in ten of twelve tasks.

Source→Target	SOURCE ONLY	DANN [26]	ARTN
BOOKS→DVD	78.7	78.4	**81.4**
BOOKS→ELECTRONICS	71.4	73.3	**75.8**
BOOKS→KITCHEN	74.5	77.9	**78.6**
DVD→BOOKS	74.6	72.3	**77.0**
DVD→ELECTRONICS	72.4	75.4	**75.9**
DVD→KITCHEN	76.5	78.3	**78.6**
ELECTRONICS→BOOKS	71.1	71.3	**72.0**
ELECTRONICS→DVD	71.9	**73.8**	73.3
ELECTRONICS→KITCHEN	84.4	85.4	**85.5**
KITCHEN→BOOKS	69.9	70.9	**72.1**
KITCHEN→DVD	73.4	**74.0**	**74.0**
KITCHEN→ELECTRONICS	83.3	84.3	**85.2**

TABLE II

Classification accuracy percentage of digits classifications among MNIST, MNIST-M, SVHN and SYN NUMS. The first row corresponds to the performance if no adaption is implemented. The proposed method outperforms the others in two of three tasks.

Method	MNIST→MNIST-M	SYN NUMS→SVHN	SVHN→MNIST
Source only	51.4	86.7	54.9
CORAL [42]	57.7	85.2	63.1
DAN [17]	76.9	88.0	71.1
DANN [26]	76.7	**91.1**	73.9
CoGAN [23]	-	-	diverge
ADDA [27]	-	-	76.0
ARTN	**81.4**	89.1	**79.4**

very difficult to train. In addition, we test our model without regularization to analyze how the regularization term of our model affects the performance. In this experiment, we evaluate the effectiveness of our approach by comparing it with DAN based on AlexNet [17], DANN [26] and the model with no adaptation.

Because of lacking sufficient images, we implement our model based on ResNet34 [3] which is pre-trained on the ImageNet dataset, and fine-tune the model on the Office-31 dataset. Different from the digits experiment, we build a residual connection for every three layers in ResNet34 instead of every layer. As for the domain classifier, we use a network with three fully connected layers. In addition, we replace the last layer of ResNet34 with a three-layer fully connected network, and use it to predict the labels of inputs. In all tasks, we employ SGD with 0.001 learning rate and the momentum is set to 0.9. We set the coefficient parameters $\lambda = 2$, and $\beta = 0.5$. The metric is prediction accuracy in the target domain. All the results are recorded after training for 70 epochs.

Results of the experiment on the Office-31 dataset are shown in Table III. In **D→A**, **W→A**, **A→W** and **A→D**, the proposed model achieves the accuracy of 59.0%, 57.5%, 75.2% and 75.7%, respectively. Thus, in all four tasks, the proposed model outperforms the best of others by 0.9%, 1.2%, 1.5% and 0.4%. In addtion, the proposed model outperforms the model with no adaptation in all tasks, which means that it avoids negative transfer learning. The results demonstrate that the proposed model can match the distributions of source and target domains properly, and the modification we make improves the performance of proposed model.

D. Regularization Analysis

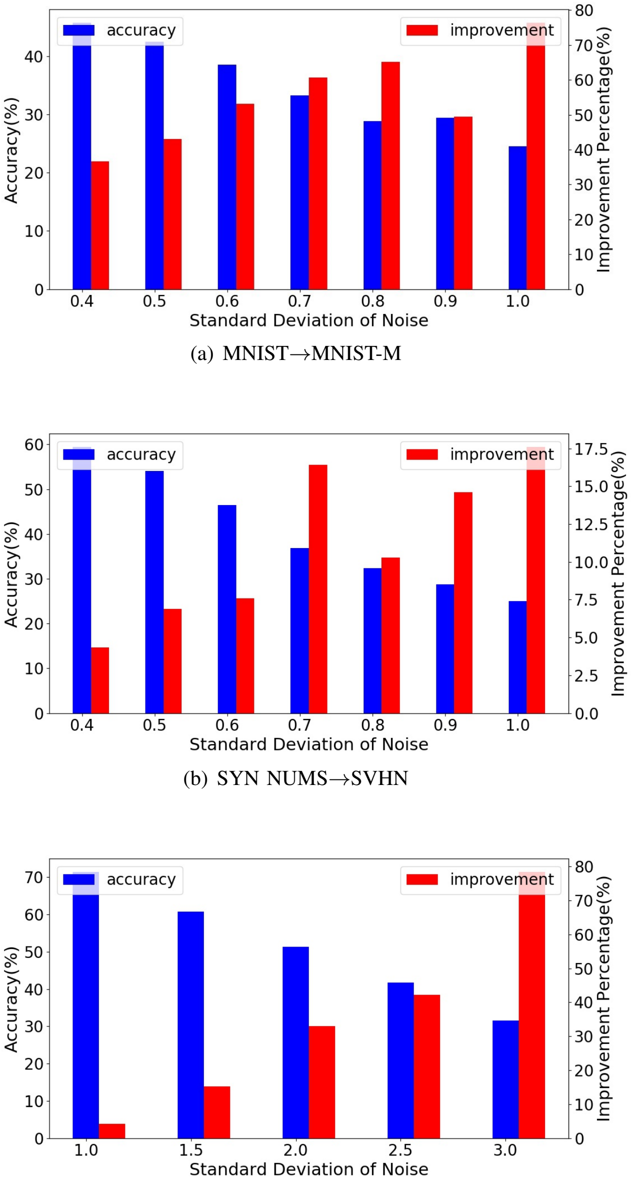
We next analyze how the regularization term of our model affects the performance. We test our model without regularization on the Office-31 dataset by setting $\beta = 0$. In this way, $\mathcal{L}$ would only consist of $\mathcal{L}_c$, $\mathcal{L}_s$ and $\mathcal{L}_t$. Except for the regularization term, this experiment has same settings as the image classification experiment does.

Results of this experiment are shown in Table III. In **D→A**, **W→A**, **A→W** and **A→D**, the proposed model without regularization achieves the accuracy of 58.1%, 56.8%, 74.7% and 73.3%, which is lower by 0.9%, 0.7%, 0.5% and 2.4% of the proposed one with such term, respectively. The model with regularization outperforms DANN and the model without regularization in all tasks, which demonstrates the effectiveness of regularization. In another word, the regularization term strengthens the generalization ability of the proposed model. It should be noted that in **D→A**, **W→A** and **A→W**, the proposed model without regularization still outperforms DANN. This means that the improvement is not only from the regularization but also the modification of its architecture.

TABLE III

CLASSIFICATION ACCURACY PERCENTAGE OF EXPERIMENT ON THE OFFICE-31 DATASET. THE FIRST COLUMN CORRESPONDS TO THE PERFORMANCE IF NO ADAPTION IS IMPLEMENTED. THE SECOND TO LAST COLUMNS CORRESPOND TO THE PERFORMANCE OF DAN, DANN, THE PROPOSED METHOD WITHOUT REGULARIZATION AND THE PROPOSED METHOD. THE PROPOSED METHOD OUTPERFORMS DAN AND DANN, AND THE REGULARIZATION TERM SHOWS A POSITIVE EFFECT ON THE PERFORMANCE IMPROVEMENT.

Method	DSLR→AMAZON	WEBCAM→AMAZON	AMAZON→WEBCAM	AMAZON→DSLR
SOURCE ONLY	57.5	55.5	68.4	68.9
DAN [17]	54.0	53.1	68.5	67.0
DANN [26]	58.1	56.3	73.7	75.3
ARTN(NO REG)	**58.1**	**56.8**	**74.7**	73.3
ARTN	**59.0**	**57.5**	**75.2**	**75.7**



(a) MNIST→MNIST-M

(b) SYN NUMS→SVHN

(c) SVHN→MNIST

Fig. 6. Accuracy of the adaptation-free method and improvement of our model in MNIST→MNIST-M, SYN NUMS→SVHN and SVHN→MNIST, where we add gaussian noise to images in the target domain. X-axis represents the standard deviation of noise, and Y-axis represents the accuracy of adaptation-free method and improvement percentage of our model in the target domain.

E. Generalization Analysis

A generalization test is taken by adding Gaussian noise to images in a target domain. In this way, the discrepancy between source and target domains is larger and discriminative information in a target domain is more difficult to capture. In this experiment, we test the anti-noise and generalization abilities of our model based on the digits experiment. For images in the source domain, we follow the settings in MNIST→MNIST-M, SYN NUMS→SVHN and SVHN→MNIST respectively, however, for images in the target domain, we add varying Gaussian noise. For MNIST→MNIST-M and SYN NUMS→SVHN, standard deviation of Gaussian noise belongs to $\{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Standard deviation in SVHN→MNIST belongs to $\{1.0, 1.5, 2.0, 2.5, 3.0\}$. The means of Gaussian noise in all tasks are 0. Results are plotted in Fig. 6. The baseline method is a model without adaptation.

Comparing the proposed model with the adaptation-free model, we can see that although noises are added to the test images, the proposed model exhibits a great advantage over the adaptation-free model. In MNIST→MNIST-M, when the standard deviation is 0.4, the accuracy of the adaptation-free model is 45.83%, whereas ours is improved by 36.6%. When standard deviation is 1.0, the accuracy of the adaptation-free model is 24.55%, whereas ours is improved by 76.4%. Similar results appear in SYN NUMS→SVHN and SVHN→MNIST, where the rate of improvement generally shows an upward trend in the case of a gradual increase of noise. Therefore, as the discrepancy between source and target domains increases, the performance advantage of the proposed model is becoming more and more obvious in comparison with a adaptation-free model. This result means that even if there exist some noises in a target domain, the proposed model can maintain excellent generalization and anti-noise abilities.

V. CONCLUSION

We propose a novel unsupervised domain adaptation model based on adversarial learning. Different from previous adversarial adaptation models which rely on extracting domain-invariant representations, our model adds a feature-shared transform network to directly map features from the source domain to the space of target features. Furthermore, we add a regularization term to help strengthen its performance. Experimental results clearly demonstrate that the proposed model can match different domains effectively.

References

[1] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[2] C. dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 69–78.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

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

[7] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151–175, 2010.

[8] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in *Advances in neural information processing systems*, 2007, pp. 137–144.

[9] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

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

[11] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2066–2073.

[12] M. Long, J. Wang, J. Sun, and S. Y. Philip, "Domain invariant transfer kernel learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 6, pp. 1519–1532, 2015.

[13] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 17–36.

[14] G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley *et al.*, "Unsupervised and transfer learning challenge: a deep learning approach," in *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop-Volume 27*. JMLR. org, 2011, pp. 97–111.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[16] A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B. K. Sriperumbudur, "Optimal kernel choice for large-scale two-sample tests," in *Advances in Neural Information Processing Systems*, 2012, pp. 1205–1213.

[17] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International Conference on Machine Learning*, 2015, pp. 97–105.

[18] M. Long, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," *arXiv preprint arXiv:1605.06636*, 2016.

[19] 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.

[20] J. Hoffman, S. Guadarrama, E. S. Tzeng, R. Hu, J. Donahue, R. Girshick, T. Darrell, and K. Saenko, "Lsda: Large scale detection through adaptation," in *Advances in Neural Information Processing Systems*, 2014, pp. 3536–3544.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[22] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: introduction and outlook," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[23] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 469–477.

[24] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," *arXiv preprint arXiv:1612.05424*, 2016.

[25] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4068–4076.

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

[27] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," *arXiv preprint arXiv:1702.05464*, 2017.

[28] Y. Chen, S. Song, S. Li, L. Yang, and C. Wu, "Domain space transfer extreme learning machine for domain adaptation," *IEEE Transactions on Cybernetics*, pp. 1–14, 2018.

[29] S. Mehrkanoon and J. A. K. Suykens, "Regularized semipaired kernel cca for domain adaptation," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2018.

[30] S. Khalighi, B. Ribeiro, and U. J. Nunes, "Importance weighted import vector machine for unsupervised domain adaptation," *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3280–3292, Oct 2017.

[31] J. Li, K. Lu, Z. Huang, L. Zhu, and H. T. Shen, "Transfer independently together: A generalized framework for domain adaptation," *IEEE Transactions on Cybernetics*, pp. 1–12, 2018.

[32] T. Kanamori, S. Hido, and M. Sugiyama, "A least-squares approach to direct importance estimation," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1391–1445, 2009.

[33] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data," in *Advances in neural information processing systems*, 2007, pp. 601–608.

[34] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *Advances in neural information processing systems*, 2008, pp. 1433–1440.

[35] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 38, no. 3, pp. 447–461, 2016.

[36] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain adaptation under target and conditional shift," in *International Conference on Machine Learning*, 2013, pp. 819–827.

[37] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2960–2967.

[38] B. Sun and K. Saenko, "Subspace distribution alignment for unsupervised domain adaptation." in *BMVC*, 2015, pp. 24–1.

[39] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1853–1865, 2017.

[40] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," in *Advances in Neural Information Processing Systems*, 2017, pp. 3733–3742.

[41] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*, 2014, pp. 647–655.

[42] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation." in *AAAI*, vol. 6, no. 7, 2016, p. 8.

[43] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Central moment discrepancy (cmd) for domain-invariant representation learning," *arXiv preprint arXiv:1702.08811*, 2017.

[44] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 539–546.

[45] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 513–520.

[46] M. Kan, S. Shan, and X. Chen, "Bi-shifting auto-encoder for unsupervised domain adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3846–3854.

[47] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, "Deep reconstruction-classification networks for unsupervised domain adaptation," in *European Conference on Computer Vision*. Springer, 2016, pp. 597–613.

[48] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *arXiv preprint arXiv:1603.06432*, 2016.

[49] M. Arjovsky and L. Bottou, “Towards principled methods for training generative adversarial networks,” *arXiv preprint arXiv:1701.04862*, 2017.

[50] M. Chen, Z. Xu, K. Weinberger, and F. Sha, “Marginalized denoising autoencoders for domain adaptation,” *arXiv preprint arXiv:1206.4683*, 2012.

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