Domain Adaptation with Invariant Representation Learning: What Transformations to Learn?

Anonymous Author(s)
Affiliation
Address
email

Abstract

Unsupervised domain adaptation, as a prevalent transfer learning setting, spans many real-world applications. With the increasing representational power and applicability of neural networks, state-of-the-art domain adaptation methods make use of deep architectures to map the input features X to a latent representation Z that has the same marginal distribution across domains. This has been shown to be insufficient for generating optimal representation for classification, and to find conditionally invariant representations, usually strong assumptions are needed. We provide reasoning why when the supports of the source and target data from overlap, any map of X that is fixed across domains may not be suitable for domain adaptation via invariant features. Furthermore, we develop an efficient technique in which the optimal map from X to Z also takes domain-specific information as input, in addition to the features X. By using the property of minimal changes of causal mechanisms across domains, our model also takes into account the domain-specific information to ensure that the latent representation Z does not discard valuable information about Y. We demonstrate the efficacy of our method via synthetic and real-world data experiments.

1 Introduction

Unsupervised domain adaptation (UDA) is a common setting for supervised learning, in which the labeled training and unlabeled test data come from different distributions. More formally, given features $X \in \mathbb{R}^d$ and labels $Y \in \mathbb{R}$, we observe labeled source and unlabeled target domain instances, represented by $(\mathbf{x}^{\mathcal{S}}, \mathbf{y}^{\mathcal{S}}) = (\mathbf{x}_k^{\mathcal{S}}, y^{\mathcal{S}})_{k=1}^{m_S}$ and $\mathbf{x}^{\mathcal{T}} = (\mathbf{x}^{\mathcal{T}})_{k=1}^{m_T}$ respectively, where $P^{\mathcal{S}}(X, Y) \neq P^{\mathcal{T}}(X, Y)$. The main challenge of domain adaptation is to use the given source domain observations for learning a predictor that will perform well in the target domain. To do so, the procedure needs to make use of some similarities between the two domains. One way to formalize this is by making assumptions about how the joint distribution joint $P(X, Y)$ changes across domains. For example, in the well-studied setting of covariate shift [32, 46, 19, 37, 2, 7], the marginal distribution $P(X)$ changes while the conditional distribution $P(Y|X)$ (i.e. the optimal predictor) is shared across domains.

However, in many real-world applications, $P(Y|X)$ can also change, and this requires making use of further assumptions. One such assumption is that the factorization $P(X, Y) = P(Y)P(X|Y)$ allows for addressing the changes in $P(Y)$ and $P(X|Y)$ independently in a situation where their respective changes are simple and easier to capture [48, 31]. In this particular setting, the problem is generally broken down into: **(1)** *Target shift:* $P(Y)$ changes across domains while $P(X|Y)$ stay the same [36, 20, 30, 48] **(2)** *Conditional shift:* $P(X|Y)$ changes across domains but $P(Y)$ stays the same [48, 27, 25]. **(3)** *Conditional-target shift:* Both $P(X|Y)$ and $P(Y)$ change independently across domains - the most general setting under this generating process assumption [48, 5]. Then, assumptions about the changes of the factors of the joint distrubution can be made so that the problem

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

is solvable, such as location-scale transformation [49, 27], or that the changing parameters lie on a low-dimensional manifold [35], and algorithms can be designed to enforce these constraints and make use of them for prediction in the target domain.

Another fruitful view of the problem is through the lens of representation learning, due to wide-spread applicability of neural architectures for many real-world problems. In particular, state of the art deep learning techniques harness the high representational capacity of neural networks to transform the input data into a latent feature representation which is predictive of the target variable Y in the source domain, and has the same distribution across domains. Formally, this means learning a function (encoder) $\phi : \mathcal{X} \to \mathcal{Z}$ from the input space to a latent space $\mathcal{Z}$, such that $P^S(Z) = P^T(Z)$. At the same time, a function $h : \mathcal{Z} \to \mathcal{Y}$ can be learnt to minimize the risk in the labeled source domainn [1]. The hope is then, that the overall function $g := h \circ \phi$ will have low prediction risk in the target domain.However, the above-described theoretical and methodological framework does not guarantee that the learnt representation Z will have any relevant information for predicting Y in the target domain. Namely, one can easily have a situation in which the learnt representation Z is *marginally invariant* ($P^S(Z) = P^T(Z)$), but not *conditionally invariant* ($P^S(Z|Y) \neq P^T(Z|Y)$), as discussed in [55]. This means that the learnt function $g := h \circ \phi$ can have very good prediction performance in the source domain, but generalize very poorly to the target domain.

In many methods, the same encoding function $\phi(X)$ across domains is used to learn invariant latent representations. This enjoys computational benefits and makes the learning procedure relatively simple, and the vast majority of approaches ([11, 22, 26, 24, 16] among many) employ this technique. However, in certain situations, the same encoding function across domains cannot learn a marginally invariant representation that is optimal for classification in the target domain. There are certain studies ([39, 40, 3]) which implement domain-specific encoders ϕ_S and ϕ_T. Unfortunately, such methods suffer from the following drawbacks: **(1)** the exact motivation behind having separate encoders for each domain is not clear; **(2)** including two separate encoders may be inefficient because it greatly increases the number of parameters that we need to learn; **(3)** in the field of UDA based on invariant representation learning, there is still no principled way to guarantee that the learned marginally invariant representation Z has sufficient structural (semantic) information, and has the potential to be conditionally invariant.

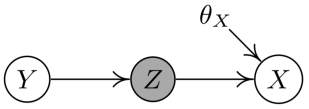
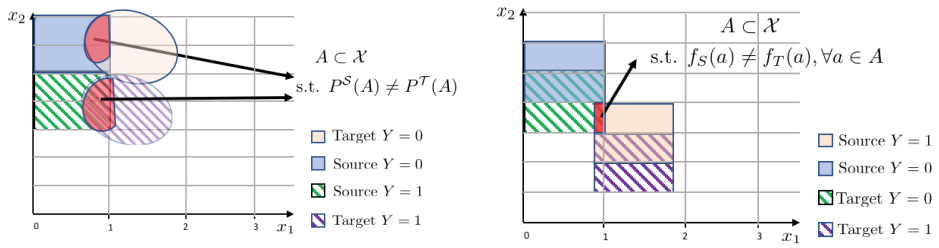


Figure 1: The underlying data-generating process under conditional shift, of the observed variables Y and X, and the latent variable Z. θ_X represents the changing parameters of $P(X|Y)$ across domains.

In this paper we assume the setting of *conditional shift*, and we make use of the data-generating process to: **(i)** justify the use of two separate encoding functions in order to infer the latent representation, **(ii)** implement the two encoding functions more efficiently, and **(iii)** constrain the latent representation Z to have meaningful structure which is useful for prediction in the target domain. In Chapter 2, we first motivate the use of two separate encoders to infer Z, via rigorous treatment and an illustrative example. Subsequently, in Chapter 3, we introduce an efficient way to implement two separate functions for inferring Z. In Chapter 3 we shall also introduce a principled way to ensure that the latent representation Z to contain useful information for prediction, and finally, in Chapter 4 we provide empirical evaluation.

1.1 Related Work

As previously mentioned, there is a vast body of work exploring approaches to make use of latent invariant representations for the purposes of unsupervised domain adaptation, using both deep learning and more traditional techniques. Namely, invariant representations were considered as a linear projection in [27], where theoretical guarantees were established regarding the conditional invariance of the latent representation and the identifiability of the joint distribution in the target domain. Furthermore, this framework was generalized to nonlinear maps to invariant latent representations, parameterized by neural network termed DANN [11]. In this framework, the constraint that $P^S(Z) = P^T(Z)$ can be enforced in various ways, including adversarial training, Maximum Mean Discrepancy [21], Wassertstein Distance [8] and Margin Disparity Discrepancy [53]. However, none of these studies have considered using two separate encoders for the two domains.



(a) (b)

Figure 2: Two different scenarios in which a single encoder $\phi(X)$ is not sufficient to learn Z.

Subsequently, this reasoning has been considered in combination with image translation (CYCADA) [18]. Another assumption frequently made in recent studies is that the input features X have a clustering structure in which each cluster has a different label Y. Therefore, further directions were pursued by incorporating additional information contained in the unlabeled target domain data, such as pseudo-labels provided by the classifier which is initially trained on the labeled source domain data [24, 43] (termed CDAN and SLPP, respectively). In addition, one can also make use of this assumption to enforce that the prediction function g does not pass through high-density regions in input space, as performed by the DIRT-T algorithm [34]. The method by [39] makes, but does not regularize their output to preserve semantic information in any way. In addition to these, approaches, the DSN algorithm [3] takes into models the changing parameters of $P(X, Y)$ in the latent representation and enforces that they are different from the invariant representation Z. The only method that we are aware of which takes domain-specific information into account is the study by [40], in which domain information is given and continuous, and therefore contains a lot more information than the discrete domain index that is typically considered. In [16], a spherical architecture is used for ϕ, however there is only one encoder for both domains. The study by [28] performs bridging between the source and the target domains using data-augmentation, but this is a technique specific to image data.

While the above-mentioned methods use various approaches to improve the quality of the learnt invariant latent representation and show competitive performance, there is no guarantee that these approaches are general enough to encompass all scenarios. Namely, the challenge that existing methods face is two-fold: **(1)** there is no principled way to ensure that marginal invariance ($P^S(Z) = P^T(Z)$) preserves conditional invariance ($P^S(Z|Y) = P^T(Z|Y)$), **(2)** there are cases (especially in lower-dimensional datasets) where the encoding function ϕ needs domain-specific information, in addition to the features X. The aim of this paper is to address these two issues.In this paper, we introduce the first invariant representation learning method that makes use of the data-generating process to justify and efficiently use domain-specific information in the encoder ϕ, and to properly constrain it to ensure that Z contains relevant label-specific information.

2 What Does it Take to Learn a Useful Invariant Representation?

To gain insights into what information is necessary for learning an intermediate representation Z via ϕ, let us consider the data-generating process depicted on Figure 1. In this graphical model, the label Y is generated first from its prior distribution $P(Y)$. Then, the invariant representation Z is generated from Y via the generating mechanism $P(Z|Y)$ and X is subsequently generated from $P(X|Z; \theta_X)$, where θ_X are the changing parameters of $P(X|Y)$ across domains. We consider θ_X as a parameter vector with a constant value for each point within a domain, and we generally assume that θ_X is low-dimensional, i.e. the change across domains in $P(X|Y)$ is *minimal*. This generating process, in part or in whole, can be exploited in many classification applications. For example, in object classification in images, Y is the label (the object the picture was taken of), X is the observed pixels, and Z can correspond to latent features with semantic relevance, such as some function of the edges and contours on the image. In this case, θ_X can correspond to different resolution, illumination conditions, or other environment-specific changes that are not relevant for predicting the class Y. In this graphical representation, we see that generally speaking, Z is conditionally dependent from θ_X given X, although they may be marginally independent. This implies that in order to recover Z, given that X is considered, the information of θ_X should also be considered in the transformation,

because it is in the Markov Blanket of Z.

To gain intuition why considering the Markov Blanket is important, consider the following scenario, depicted in Figure 2a. In this setting, we are given the source domain with Gaussian distribution, and the target domain with Uniform distribution. Here, we have a region $A \subset \mathcal{X}$ (highlighted in red), in which both of the domains' distributions have support, so there is support overlap. However, each domain has a different density in this region. The following proposition claims that in this situation, we cannot infer an invariant representation Z from X with a single encoder ϕ:

Proposition 1: *Let $A \subset \mathcal{X}$ be a region in input space such that $P^{\mathcal{S}}(X \in A) > 0$ and $P^{\mathcal{T}}(X \in A) > 0$. Furthermore, let $\phi : \mathcal{X} \to Z$ be an encoder s.t. $A = \{x : \phi(x) \in B\}$ for some $B \subset \mathcal{Z}$. Then, there is no function ϕ s.t. $P^{\mathcal{S}}(\phi(X) \in B) = P^{\mathcal{T}}(\phi(X) \in B)$.*

All omitted proofs can be found in the supplementary materials. This proposition implies that if the supports of the marginal distributions of the source and target domains overlap, and if the distributions are different in the specific region of the support overlap (set A), then we cannot use a fixed transformation to transform the data in that region to an invariant representation in $\mathcal{Z}$ space.

There is an additional case in which a single encoder ϕ is not sufficient to learn an optimal latent representation Z. The scenario is illustrated on Figure 2b. In this case, even though both domains have the same distribution $P(X)$ in the region of support overlap between the source and the target domain (labeled in red), the value of Y is opposite between the two domains. The following proposition demonstrates that in this setting, a single encoding function ϕ is inherently sub-optimal:

Proposition 2: *Let the true labeling functions in the source and target domain be $f_S, f_T : \mathcal{X} \to \mathcal{Y}$, respectively. Let $A \subset \mathcal{X}$ be a region s.t. $f_S(a) \neq f_T(a), \forall a \in A$. Let $g : \mathcal{X} \to \mathcal{Y}$ be a composition of a representation learner $\phi : \mathcal{X} \to \mathcal{Z}$ and a classifier $h : \mathcal{X} \to \mathcal{Y}$. If ϕ is the same function across domains, then for a 0-1 loss, the risk over the region A: $\epsilon^A(g) = \epsilon^A(g)_S + \epsilon^A(g)_T \geq 1$.*

In this scenario, no matter what criterion one uses to find the representation for downstream classification using h, as long as the two domains use the same transformation ϕ for the representation, the points in A cannot be correctly classified in the target domain, if they are correctly classfied in the source domain. This is because the final classifer of the data, $g = h \circ \phi$, is fixed across domains. As a consequence, any point $a \in A$ will be mapped to the same class label, no matter which domain it is in. These two scenarios generally occur in low-dimensional datasets, such as scientific and healthcare applications. In very high-dimensional structured data such as images and text, there will be some domain-specific dimensions which will have very different values in the source and the target domain (such as brightness in images in daytime vs. nighttime). This will result in no overlap between the supports of the source and target domain distributions, and thus the domain-specific dimensions will act as domain specific information. In this case using ϕ which is only a function of X is sufficient. However, even in this scenario, we need to find a principled way to constrain the encoder such that Z contains relevant label-specific information for the prediction in the target domain.

The observations from the perspective of the causal mechanism of generating X from Z and θ_X provide hints how to tackle the problem of UDA using invariant latent representations in a general way. First of all, we observe that θ_X needs to be an input to the encoder ϕ, in addition to X. Secondly, in this causal mechanism, we can safely assume that the influence of θ_X on the relationship between X and Z (and therefore the relationship between X and Y) is minimal. This allows us to: **(1)** model θ_X as a latent variable and provide it is an input to $\phi(X, \theta_X)$, instead of learning two separate encoders $\phi_{\mathcal{S}}$ and $\phi_{\mathcal{T}}$; **(2)** mimic the causal mechanism of generating X, given by $X = \tilde{\phi}(Z, \theta_X)$, via a decoder $\tilde{\phi} : \mathcal{Z} \times \Theta \to \mathcal{X}$, in which we constrain the influence of θ_X to be *minimal*. The hope is then that such a decoder would act as a regularizer on the encoder ϕ, forcing it to preserve important semantic information when inferring Z. In this paper, we design a method based on this reasoning in order to address the problem of UDA in a principled manner.

3 Proposed Method: Domain-Specific Adversarial Training

Motivated by the above discussion, we aim to design a domain-adversarial network which can infer the latent domain-specific parameters θ_X. Let $\theta_X \in \{\theta_X^{\mathcal{S}}, \theta_X^{\mathcal{T}}\}$, which are its possible values in the source and target domains. Our goal is to design a neural network which can provide good estimates for them, given by $\hat{\theta}_X^{\mathcal{S}}$ and $\hat{\theta}_X^{\mathcal{T}}$, and subsequently make use of them for learning an appropriate invariant representation $Z = \phi(X, \theta_X)$ which is predictive of Y. The model for the proposed method DSAN (Domain-Specific Adversarial Network) is depicted in Figure 3.

We first consider the domain index of the data: $j \in \{\mathcal{S}, \mathcal{T}\}$ (which indicates whether a data-point comes from the source or target domain), and parameterize the estimated latent domain-specific parameters $\hat{\theta}_X^{(j)}$ as a function of the domain index of the i-the data-point j_i (0 for source, 1 for target domain). We make use of $\hat{\theta}_X^{(j)}$ and data encoder ϕ to obtain $\mathbf{z}_i^{(j)} = \phi(\mathbf{x}_i^{(j)}, \hat{\theta}_X^{(j)})$, where $\mathbf{x}_i^{(j)}$ and $\mathbf{z}_i^{(j)}$ represent the the i-th input point and latent representation point respectively, from the j-th domain. We can then use the latent representation $\mathbf{z}_i^{(j)}$ to obtain a softmax probability for the predicted label label $\tilde{y}_i^{(j)} = h(\mathbf{z}_i^{(j)})$, and this predictor can be trained in the source domain. To enforce the condition $P^S(Z) = P^T(Z)$, we also have an adversarial predictor which predicts the domain index, given by $\tilde{j} = h_a(\mathbf{z}_i^{(j)})$. In order to successfully infer the latent changes θ_X, our model also needs to make use of (or mimic) the data-generating process depicted on Figure 1, in particular the process of generating X from Z and θ_X. This can be achieved via a decoder $\tilde{\phi}$ which reconstructs the features X in both the source and the target domain, from Z and θ_X, given by $\tilde{\mathbf{x}}_i^{(j)} = \tilde{\phi}(\mathbf{z}_i^{(j)}, \hat{\theta}_X^{(j)})$. As described so far, our model consists of the following loss functions:

$$\mathcal{L}_{reconst.} = \frac{1}{m_S + m_T} \sum_{i=1}^{m_S+m_T} ||\mathbf{x}_i - \tilde{\mathbf{x}}||^2, \; \mathcal{L}_{classif.} = -\frac{1}{m_S} \sum_{c=1}^{C} \sum_{i=1}^{m_S} y_{ic} log(\tilde{y}_{ic}),$$

$$\mathcal{L}_{inv.} = -\sum_{i=1}^{m_S+m_T} \{j_i \log \tilde{j}_i + (1 - j_i) \log(1 - \tilde{j}_i)\}, \; \mathcal{L}_{cent.} = -\sum_{i=1}^{m_T} h(\mathbf{z}_i^{\mathcal{T}})^T \log(h(\mathbf{z}_i^{\mathcal{T}}))$$

Here, $\mathcal{L}_{reconst.}$, $\mathcal{L}_{reconst.}$, and $\mathcal{L}_{inv.}$ are the reconstruction, classification and adversarial loss respectively. $\mathcal{L}_{cent.}$ is the conditional cross-entropy of the predictions in the target domain, and intuitively, it serves to enforce that the decision boundary does not cross data-dense regions [34]. The current loss functions do not involve a constraint on the encoding transformation ϕ which would prevent it from discarding valuable information about Y. We now describe how we can make use of the data-generating process and the assumption of minimal change across domains to enforce such a constraint.

3.1 Enforcing Minimal Change by Mutual Information Minimization

Before we delve into the details of how to enforce minimality of the influence of θ_X on the generating process of X from Y, we need to find a way to measure it. One way to formally do so is the joint mutual information $I(\theta_X; (X, Y))$. Conveniently, this term is related to a distributional divergence measure that can be minimized, as stated by the following lemma:

Lemma 1: *Let $X, Y \sim P(X, Y; \theta_X)$, and let θ_X be a discrete random variable with possible values $\theta_X^{\mathcal{S}}$ and $\theta_X^{\mathcal{T}}$ in the source and target domains respectively, with a uniform prior $P(\theta_X = \theta_X^{\mathcal{S}}) = 0.5$, $P(\theta_X = \theta_X^{\mathcal{T}}) = 0.5$. Then, $I(\theta_X; (X, Y)) = JSD(P_{X,Y;\theta_X=\theta_X^{\mathcal{S}}} || P_{X,Y;\theta_X=\theta_X^{\mathcal{T}}})$, where JSD is the Jensen-Shannon Divergence, and $P_{X,Y;\theta_X=\theta_X^{\mathcal{S}}}$ and $P_{X,Y;\theta_X=\theta_X^{\mathcal{T}}}$ are the source and target domain joint distributions respectively.*

Therefore, to measure the influence of θ_X on the generating process as mimicked by our decoder, we need to measure the JSD between the joint distributions as implied by the reconstructed data, given by JSD$(P_{X,Y;\theta_X=\hat{\theta}_X^{\mathcal{S}}} || P_{X,Y;\theta_X=\hat{\theta}_X^{\mathcal{T}}})$ (note that here we are using inferred domain-specific variables $\hat{\theta}_X^{\mathcal{S}}$ and $\hat{\theta}_X^{\mathcal{T}}$). In theory, given enough data and expressiveness of the decoder, we can assume that we can learn to reconstruct the data perfectly. However, due to lack of labeled data in the target domain, we can only reconstruct the unlabeled features in the target domain, and have no access to $P_{X,Y;\theta_X=\hat{\theta}_X^{\mathcal{T}}}$). To cope with this, we can use the inferred domain-specific variable $\hat{\theta}_X^{\mathcal{T}}$ to

translate the source domain data to the target domain via the decoder: $\tilde{\mathbf{x}}^{\mathcal{S}}_{trans.} = \tilde{\phi}(\mathbf{z}^{\mathcal{S}}, \hat{\theta}^{\mathcal{T}}_X)$. The following Theorem, adopted from [14], gives an upper bound of the JSD which is easy to minimize:

Theorem 1 [14]: *Let $P_{YX|\theta_X=\theta^S}$ and $P_{YX|\theta_X=\theta^\mathcal{T}}$ denote the source and target domain distributions respectively. Let $Q^{h_c}_{Y|X}$ denote the conditional distribution of Y given X specified by the auxiliary classifier h_c. We have:*

$$\begin{aligned} JSD(P_{XY;\theta_X=\theta^{\mathcal{S}}_X}||P_{XY;\theta_X=\theta^{\mathcal{T}}_X}) &\leq 2c_1\sqrt{JSD(P_{X|\theta_X=\theta^{\mathcal{S}}_X}||P_{X|\theta_X=\theta^{\mathcal{T}}_X})} \\ &+ c_2\sqrt{KL(P_{Y|X,\theta_X=\theta^{\mathcal{S}}_X}||Q^{h_c}_{Y|X})} + c_1\sqrt{KL(P_{Y|X,\theta_X=\theta^{\mathcal{T}}_X}||Q^{h_c}_{Y|X})} \end{aligned}$$

Here, $\text{JSD}(P_{X|\theta_X=\hat{\theta}^{\mathcal{S}}_X}||P_{X|\theta_X=\hat{\theta}^{\mathcal{T}}_X})$ is equal to the JSD between the source and target domain marginal distributions of X and is fixed, we can ignore this term when we minimize the upper bound using our algorithm. Therefore, to minimize the two KL divergences on the RHS:

$$\text{KL}(P_{Y|X,\hat{\theta}_X=\theta^{\mathcal{S}}_X}||Q^{h_c}_{Y|X}) \text{ and } \text{KL}(P_{Y|X,\theta_X=\hat{\theta}^{\mathcal{T}}_X}||Q^{h_c}_{Y|X}),$$

it suffices to train a joint auxiliary classifier h_c on the reconstructed source domain data $\tilde{x}^{\mathcal{S}}$ and the translated data $\tilde{x}^{\mathcal{T}}_{trans.}$. Therefore, we introduce a cross-entropy loss $\mathcal{L}^{trans.}_{inv.}$ for the auxiliary classifier, whose two terms can be used to minimize the two KL divergences respectively:

$$\mathcal{L}^{trans}_{inv.} = -\sum_{j=1}^{C}\sum_{i=1}^{m_S} y_{ij}\log(\tilde{y}^c_{ij}) - \sum_{j=1}^{C}\sum_{i=1}^{m_S} y_{ij}\log(\tilde{y}^c_{ij,trans.}), \text{ where } \tilde{y}^c_i = h_c(\tilde{x}^{\mathcal{S}}_i),$$

and $\tilde{y}^c_{i,trans.} = h_c(\tilde{x}^{\mathcal{S}}_{i,trans.})$ are the softmax predictions of the reconstruction and the translation of the source domain respectively, using predictor h_c. Therefore, the total loss function can be formulated in the following manner: $\mathcal{L}_{total} = \lambda_\alpha\mathcal{L}_{reconst.} + \lambda_\delta\mathcal{L}_{classif.} - \lambda_\gamma\mathcal{L}_{inv.} + \lambda_\tau\mathcal{L}^{trans.}_{inv.}$, and can be minimized by alternating optimization:

$$\min_{\phi,\tilde{\phi},\rho,h,h_c} \mathcal{L}_{total}$$
$$\min_{h_a} \lambda_\gamma\mathcal{L}_{inv.}.$$

This alternating minimization problem can be solved using Gradient-Reversal Layers (GRL), as shown in [12]. $\mathcal{L}_{inv.}$ can also be expressed in terms of Maximum Mean Discrepancy (MMD) [15]. Our model is now equipped with a way to ensure that when reconstructing the data, θ_X cannot play a major role in the decoder $\tilde{\phi}$. Since θ_X correspond to minimal changes of the conditional distribution $P(X|Y)$, this forces Z to retain Y-specific structure, and in turn implicitly regularizes the encoder ϕ, resulting in a conditionally invariant representation. In the following section, we will empirically demonstrate this property of the constraint.

4 Empirical Evaluation

In order to provide intuition about the proposed method's advantages in the challenging scenarios described above, we first designed a simple but informative 2D simulated example for domain adaptation using mixtures of Gaussian distributions. This simulated dataset is presented on Figure 4(a), and is representative of the example discussed in Figure 2, since there is a significant region of overlap between the two domains, in which the points have opposite labels across domains (details about the dataset can be found in the supplementary materials). In this experiment, we compared our method DSAN, with the following baselines: (1) DSAN-unreg., which is our proposed method but without regularizing the transformation ϕ using the mutual information minimization in our loss; (2) DSN, the method proposed in [3], which makes use of domain-specific private encodings, but which are only used for reconstruction of the data, and not for inferring the shared invariant representation Z; (3) DANN, the classical domain-adversarial method proposed by [12]. We used a 2D hidden representation Z (more details about implementation and tuning can be found in the supplement).

The results are presented in Table 3, demonstrating that our approach greatly outperforms

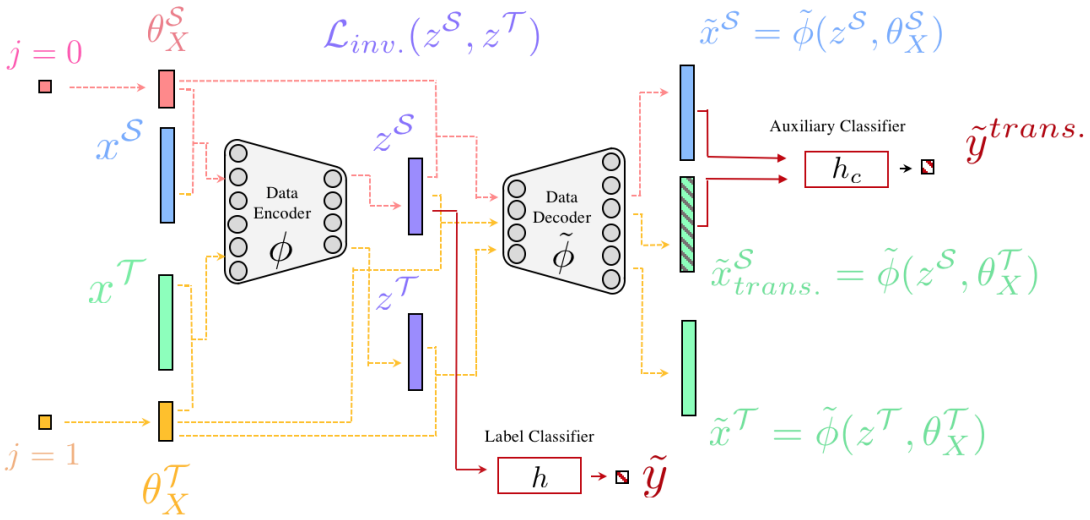
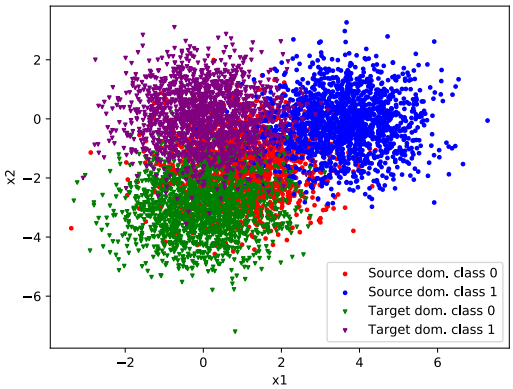


Figure 3: A diagram of the proposed autoencoder framework. Here, the domain index is mapped to θ_X. Subsequently, the input data $\mathbf{x}$ and θ_X are mapped via ϕ to a latent representation $\mathbf{z}$, which in turn is reconstructed $\tilde{\mathbf{x}}$ using the decoder $\tilde{\phi}$, and θ_X as additional input. In addition, the penalty $\mathcal{L}_{inv.}$ is enforced to ensure invariance between Z^S and Z^T. The hidden representation is also used to predict $\tilde{\mathbf{y}}$ in the source domain. Finally, $\mathcal{L}_{inv.}^{trans-y}$ enforces the minimal change of $P(X|Y)$ across domains. The pink and orange lines depict the flow of the source and target data and domain-specific information respectively.

the baselines. To visualize how our algorithm is able to solve this challenging example with much greater success, we also present the 2D invariant representations in the source and target domains on Figure 5. At the top, we have the hidden representations Z of DSAN-unreg., and on the bottom the ones yielded by our method DSAN. On the left side we present Z in the source domain, labeled by the true values, in the middle we show this representation in the target domain labeled with the prediction of the algorithm, and on the right-hand side we present Z in the target domain labeled with the true labels. From this figure, one can appreciate that the unregularized version of our approach yields a representation Z which discards a lot of information about Y, as can be seen in parts 5b and 5c. On the other hand, parts 5e and 5f show that our algorithm with the mutual information minimization constraint can successfully preserve label-specific information in the invariant representation.

4.1 Real Data Experiments

To evaluate our method on real datasets, we consider three datasets and respective tasks from various domains of applications: cross-domain Wi-Fi localization, Amazon product reviews and image classification. For detailed descriptions of the experimental design, hyperparameter tuning and neural network architectures, as well as ablation studies, we refer the interested reader to the supplementary materials. We performed evaluation on the following datasets: **Wi-Fi localization**: Introduced in [50], is a dataset in which wireless signal data was collected in a hallway area discretized as a grid. In each grid, data was recorded from 67 access points (dimensions). The task is to predict the location from the signals (which has been converted to a classification problem with 19 classes). There are three domains in this dataset (with 1140 points each), collected in three different time points. This is a low-dimensional dataset, in which we verified via manual inspection and tSNE and PCA projections that both scenarios of support overlap described in Chapter 2 hold (plots are provided in the supplement); **Amazon Review:** Amazon Review is another benchmark for multi-domain sentiment analysis. It contains positive and negative reviews of four kinds of products: Kitchen appliance (K), DVDs (D), Electronics (E), and Books (B). **ImageCLEF:** A standard UDA benchmark dataset for image classification, consisting of three domains: *Caltech-256*(C), *ImageNet ILSVRC*(I), and *Pascal VOC2012*(P), consisting of 12 classes. **Baselines:** We compare DSAN with classical approaches like TCA [29] and GFK [13], as well as with some deep transfer learning models. Three recently proposed methods, DSR [4], DIRT-T [33], MDD [52], BSP [6], MSTN [44] and RSDA [16] are included. For both the real-world and simulated experiments, we ran and tuned the baseline



(a) Scatter plot of simulated data

(b) Accuracy (%) on ImageCLEF dataset for unsupervised domain adaptation (reproduced by [16]

)

Models	I → P	P → I	I → C	C → I	C → P	P → C	Average
DResNet-50 [17]	74.8	83.9	91.5	78.0	65.5	91.2	80.7
iCAN [51]	79.5	89.7	94.7	89.9	78.5	92.0	87.4
CDAN [23]	77.7	90.7	97.7	91.3	74.2	94.3	87.7
SymNets [54]	80.2	93.6	97.0	93.4	78.7	96.4	89.9
SAFN+ENT [45]	79.3	93.3	96.3	91.7	77.6	95.3	88.9
CAT [9]	77.2	91.6	95.5	91.3	75.3	93.6	87.3
DANN [11]	75.0	86.0	96.2	87.0	74.3	91.5	85.0
DANN+S [16]	78.3	91.0	96.8	91.8	77.7	95.2	88.5
RSDA-DANN [16]	79.2	93.0	**98.3**	93.6	78.5	**98.2**	90.1
MSTN [44]	77.3	91.3	96.8	91.2	77.7	95.0	88.2
DSAN-U	79.4	94.5	96.8	94.5	76.1	88.5	88.3
DSAN	**81.0**	**95.7**	**97.3**	**95.3**	**80.5**	**97.0**	**91.1**

Figure 4

methods ourselves (all baselines for Wi-Fi and simulated, and MDD and DIRT-T for Amazon dataset).

The classification accuracies on the Amazon-Review Dataset for unsupervised domain adaptation are shown in Table 2, which shows that our algorithm model outperforms the baselines in the vast majority of source-target directions. We also perform a Wilcoxon signed-rank test on the experiment result of different random seed, and the p-value is 0.0216. The performance of our method on the Wi-Fi localization dataset is shown on Table 1. In this experiment, we sub-sampled 950 points in each domain to create 10 replicate experiments, and we present the accuracies and standard deviations across all replicates. From the results, one can appreciate that our method outperforms all baselines, for the majority of the pairs. For one of the remaining two pairs in which it does not, the performance is within margin of error with the strongest baseline. We calculated a Wilcoxon p value $p = 0.002$ across all replicate experiments combined. The performance of our method on the ImageCLEF dataset is presented on Table 4(b), in which one can appreciate that we achieve state-of-the-art performance on all transfer directions.

5 Conclusion

In this paper, we have demonstrated that the existing paradigm of representation learning for domain adaptation using a fixed encoder ϕ of the data, can be sub-optimal when the same points can have different labels across domains. We have introduced a method which takes domain-specific information in order to provide the data encoder with the necessary flexibility and appropriate constraints, in order to retain valuable information about Y in the latent representation Z. We note that in many real-world applications with high-dimensional datasets, the supports of the source and target domains may not overlap. In this case, domain-specific information is contained in the features X and can be automatically used by the encoder $\phi(X)$. However, $\phi(X)$ would then have the same problem with excessive flexibility and can still suffer from finding trivial reprepresentatons of Z that discard information about Y. Therefore, our principle for constraining domain-specific encoders would still be beneficial in this scenario. For future work, a promising direction is combining this framework motivated by the data-generating process with more powerful encoders, such as the spherical neural network proposed by [16].

Table 1: Accuracy (%) on Wi-Fi localization dataset for unsupervised domain adaptation

Models	t1 → t2	t1 → t3	t2 → t1	t2 → t3	t3 → t1	t3 → t2	Average
DANN [11]	31.30((1.3)	33.15(1.9)	*38.03(2.0)*	32.41(2.3)	28.54((0.8)	31.12(1.6)	32.59
DSN [3]	31.25(1.5)	34.23(2.4)	34.32(2.8)	29.47(0.7)	27.0(1.6)	31.8(1.3)	31.35
DSR [4]	34.20(1.3)	34.62(4.8)	30.61(3.5)	34.72(1.1)	33.13(0.5)	31.61(0.9)	31.13
MDD [52]	30.92(2.1)	34.42(1.8)	30.01(1.5)	27.58(1.8)	33.25(1.0)	27.34(2.3)	30.59
DIRT-T [33]	36.38(1.2)	33.66(2.7)	37.12(3.4)	31.85(2.87)	33.98(1.42)	28.48(2.68)	33.58
BSP [33]	35.5(1.3)	32.16(2.1)	31.15(1.4)	31.06(1.9)	32.91(1.1)	33.83(2.2)	32.72
DSAN-U	31.16(1.1)	29.01(1.7)	32.76(0.9)	32.18(1.1)	31.02(1.7)	**33.62(1.8)**	31.45
DSAN	**40.45(1.6)**	**37.35(2.3)**	**38.22(2.4)**	**39.94(2.1)**	**36.47(3.5)**	**34.18(2.9)**	**37.77**

Table 2: Accuracy (%) on Amazon Review dataset for unsupervised domain adaptation

Methods	B→D	B→E	B→K	D→B	D→E	D→K	E→D	E→B	E→K	K→B	K→D	K→E	Avg
NN	49.6	49.8	50.3	53.3	51.0	53.1	50.8	50.9	51.2	52.2	51.2	52.3	51.3
TCA [29]	63.6	60.9	64.2	63.3	64.2	69.1	59.5	62.1	74.8	64.1	65.4	74.5	65.5
GFK [13]	66.4	65.5	69.2	66.3	63.7	67.7	62.4	63.4	73.8	65.5	65.0	73.0	66.8
SA [10]	67.0	70.8	72.2	67.5	67.1	69.4	61.4	64.9	70.4	64.4	64.6	68.2	67.3
BDA [41]	64.2	62.1	65.4	62.4	66.3	68.9	59.6	61.6	74.7	62.7	64.3	74.0	65.5
CORAL [38]	71.6	65.1	67.3	70.1	65.6	67.1	67.1	66.2	77.6	68.2	68.9	75.4	69.1
JGSA [47]	66.6	75.0	72.1	55.5	67.3	65.6	51.6	50.8	55.0	58.3	56.4	51.7	60.5
DANN [11]	78.4	73.3	77.9	72.3	75.4	78.3	71.3	73.8	85.4	70.9	74.0	84.3	64.9
MDD [52]	77.1	74.4	77.0	74.7	74.1	76.3	72.4	70.2	83.3	69.3	73.2	82.8	75.4
DIRT-T [33]	78.6	76.1	75.5	76.8	75.2	79.1	69.6	71.0	84.2	69.2	73.3	79.5	75.7
EasyTL [42]	79.8	79.7	80.9	79.9	80.8	82.0	75.0	**75.3**	84.9	**76.5**	76.3	82.5	79.5
BSP [6]	79.33	73.86	75.86	75.67	74.77	77.2	72.81	71.3	84.02	70.92	73.59	84.29	76.14
RSDA [16]	80.1	77.8	84.4	76.6	79.4	82.3	73.6	71.1	86.6	71.8	74.7	84.4	77.9
DSAN-U	81.0	78.6	78.9	78.4	79.4	83.3	76.2	74.5	87.4	73.1	77.0	83.0	78.9
DSAN	**82.7**	**80.8**	**82.6**	79.5	**81.4**	**85.3**	**76.7**	75.1	**88.0**	73.8	**77.3**	**85.0**	**80.7**

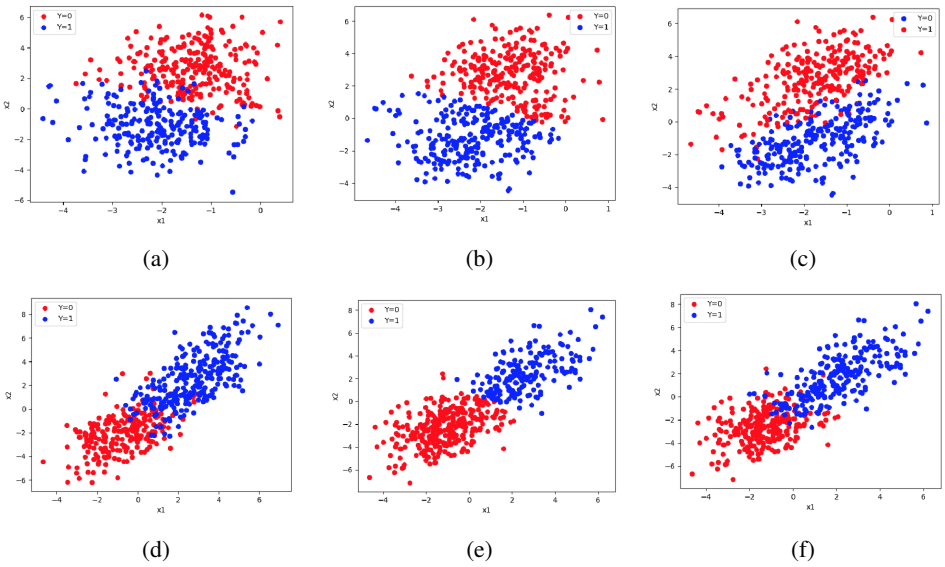


Figure 5:
Z in the simulations. First three scatter-plots show it withour regularization: (a) Z of the source domain labeled with true labels, (b) Z of the target domain labeled with the predictions (c) Z of the target domain with the true labels. (d), (e), and (f) show the same, but with regularization.

Table 3: Results on the simulated data.

Model	DANN	DSN	DSAN-unreg.	DSAN
Accuracy	58.2(4.3)	58.8(5.2)	79.4(2.2)	**87(1.0)**

References

[1] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Proceedings of NIPS 2006*, 2007.

[2] S. Ben-David, S. Shalev-Shwartz, and R. Urner. Domain adaptation–can quantity compensate for quality? In *ISAIM 2012*, 2012.

[3] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016.

[4] R. Cai, Z. Li, P. Wei, J. Qiao, K. Zhang, and Z. Hao. Learning disentangled semantic representation for domain adaptation. In *IJCAI: proceedings of the conference*, volume 2019, page 2060. NIH Public Access, 2019.

[5] Q. Chen, Y. Liu, Z. Wang, I. Wassell, and K. Chetty. Re-weighted adversarial adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7976–7985, 2018.

[6] X. Chen, S. Wang, M. Long, and J. Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *International conference on machine learning*, pages 1081–1090. PMLR, 2019.

[7] C. Cortes, Y. Mansour, and M. Mohri. Learning bounds for importance weighting. In *NIPS 23*, 2010.

[8] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *NIPS*, 2017.

[9] Z. Deng, Y. Luo, and J. Zhu. Cluster alignment with a teacher for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9944–9953, 2019.

[10] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 2960–2967, 2013.

[11] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[12] 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*, 17(1):2096–2030, 2016.

[13] 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*, pages 2066–2073. IEEE, 2012.

[14] M. Gong, Y. Xu, C. Li, K. Zhang, and K. Batmanghelich. Twin auxilary classifiers gan. In *Advances in Neural Information Processing Systems*, pages 1328–1337, 2019.

[15] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *JMLR*, 13:723–773, 2012.

[16] X. Gu, J. Sun, and Z. Xu. Spherical space domain adaptation with robust pseudo-label loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9101–9110, 2020.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. PMLR, 2018.

[19] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *NIPS 19*, pages 601–608, 2007.

[20] Z. C. Lipton, Y. Wang, and A. Smola. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916*, 2018.

[21] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *ICML-15*, pages 97–105, 2015.

[22] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[23] M. Long, Z. Cao, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. *arXiv preprint arXiv:1705.10667*, 2017.

[24] M. Long, Z. Cao, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650, 2018.

[25] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207, 2013.

[26] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2208–2217. JMLR. org, 2017.

[27] T. L. D. T. C. G. M. Gong, K. Zhang and B. Schölkopf. Domain adaptation with conditional transferable components. In *ICML 2016*, 2016.

[28] J. Na, H. Jung, H. Chang, and W. Hwang. Fixbi: Bridging domain spaces for unsupervised domain adaptation. *arXiv preprint arXiv:2011.09230*, 2020.

[29] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

[30] M. D. Plessis and M. Sugiyama. Semi-supervised learning of class balance under class-prior change by distribution matching. In *ICML-12*, pages 823–830, 2012.

[31] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij. On causal and anticausal learning. In *ICML-12*, Edinburgh, Scotland, 2012.

[32] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244, 2000.

[33] R. Shu, H. Bui, H. Narui, and S. Ermon. A DIRT-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018.

[34] R. Shu, H. H. Bui, H. Narui, and S. Ermon. A dirt-t approach to unsupervised domain adaptation. In *Proc. 6th International Conference on Learning Representations*, 2018.

[35] P. Stojanov, M. Gong, J. G. Carbonell, and K. Zhang. Data-driven approach to multiple-source domain adaptation. *Proceedings of machine learning research*, 89:3487, 2019.

[36] A. Storkey. When training and test sets are different: Characterizing learning transfer. In J. Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, editors, *Dataset Shift in Machine Learning*, pages 3–28. MIT Press, 2009.

[37] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60:699–746, 2008.

[38] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[39] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[40] H. Wang, H. He, and D. Katabi. Continuously indexed domain adaptation. *arXiv preprint arXiv:2007.01807*, 2020.

[41] J. Wang, Y. Chen, S. Hao, W. Feng, and Z. Shen. Balanced distribution adaptation for transfer learning. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1129–1134. IEEE, 2017.

[42] J. Wang, Y. Chen, H. Yu, M. Huang, and Q. Yang. Easy transfer learning by exploiting intra-domain structures. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1210–1215. IEEE, 2019.

[43] Q. Wang and T. P. Breckon. Unsupervised domain adaptation via structured prediction based selective pseudo-labeling. *arXiv preprint arXiv:1911.07982*, 2019.

[44] S. Xie, Z. Zheng, L. Chen, and C. Chen. Learning semantic representations for unsupervised domain adaptation. In *International conference on machine learning*, pages 5423–5432. PMLR, 2018.

[45] R. Xu, G. Li, J. Yang, and L. Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1426–1435, 2019.

[46] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *ICML-04*, pages 114–121, Banff, Canada, 2004.

[47] J. Zhang, W. Li, and P. Ogunbona. Joint geometrical and statistical alignment for visual domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1859–1867, 2017.

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

[49] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In *ICML-13*, 2013.

[50] K. Zhang, V. Zheng, Q. Wang, J. Kwok, Q. Yang, and I. Marsic. Covariate shift in hilbert space: A solution via sorrogate kernels. In *International Conference on Machine Learning*, pages 388–395, 2013.

[51] W. Zhang, W. Ouyang, W. Li, and D. Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3801–3809, 2018.

[52] Y. Zhang, T. Liu, M. Long, and M. Jordan. Bridging theory and algorithm for domain adaptation. volume 97 of *Proceedings of Machine Learning Research*, pages 7404–7413, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

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

[54] Y. Zhang, H. Tang, K. Jia, and M. Tan. Domain-symmetric networks for adversarial domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5031–5040, 2019.

[55] H. Zhao, R. T. d. Combes, K. Zhang, and G. J. Gordon. On learning invariant representation for domain adaptation. *arXiv preprint arXiv:1901.09453*, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes]
- Did you include the license to the code and datasets? [No] proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]