Domain Adaptation with Adversarial Neural Networks and Auto-encoders

Han Zhao

Monday 8th May, 2017

Background. Domain adaptation focuses on the situation where we have data generated from multiple domains, which are assumed to be different, but similar, in a certain sense. In this work we focus on the case where there are two domains, known as the source and the target domain. The source domain is assumed to have a large amount of labeled data while labeled data in the target domain is scarce. The goal of domain adaptation algorithms is to generalize better on the target domain by exploiting the large amount of labeled data in the related source domain.

Aim. In this data analysis project we seek to develop a neural network model that is able to learn feature representations which are invariant to the shift between source and target domains.

Methods. Our model is based on the recently proposed domain adversarial neural network, where the goal is to learn representations that are discriminative for the main learning task at the source domain, while at the same time being invariant to the shift between the source and target domains. Our model improves over the domain adversarial neural network by incorporating an auxiliary auto-encoder that works as an unsupervised regularizer to help the training of representations.

Results. We conduct experiments on the Amazon benchmark data set for sentiment analysis to compare the proposed model with related approaches in the literature. We show that our model significantly outperforms alternative approaches in classification accuracies.

Conclusions. This paper explores to combine two strands of domain adaptation methods, i.e., the representation based adversarial training, and the unsupervised pretraining scheme using auto-encoders, into a unified framework. The proposed method is able to combine the advantages of the above two complementary methods, leading to better generalization on the target domain.

Keywords: domain adaptation, deep learning, adversarial learning, auto-encoder.

1 Introduction

In the standard setting of computational learning theory (Valiant, 1984; Vapnik and Vapnik, 1998), both training and test instances are assumed to be drawn from the same underlying distribution. In practice, making accurate predictions relies heavily on the existence of labeled data for the desired tasks. However, generating labeled data for new learning tasks is often time-consuming. As a result, this poses an obstacle for applying machine learning methods to broader application domains. It is thus desirable to develop methods that can exploit data from multiple related domains. Such a scenario is known as *domain adaptation*, the main topic of this analysis project. Domain adaptation focuses on the situation where we have data generated from multiple domains, which are assumed to be different, but similar, in a certain sense. In this project we focus on the case where there are two domains, known as the source and the target domain. The source domain is assumed to have a large amount of labeled data while labeled data in the target domain is scarce. The goal of domain adaptation algorithms under this setting is to generalize better on the target domain by exploiting the large amount of labeled data in the related source domain.
In this project we are going to attack the domain adaptation problem by proposing a unified framework based on the recent advances in adversarial neural networks. One way to approach domain adaptation is to develop invariant feature representations that have similar marginal distributions on both source and target domains (Ajakan et al., 2014; Ganin and Lempitsky, 2015; Ganin et al., 2016). Another strand of research in unsupervised learning that motivates us is to build robust feature representations by stacked denoising auto-encoders (Vincent et al., 2008; Chen et al., 2012). Both approaches have been demonstrated to be effective in exploiting unlabeled instances from target domains to help generalization via training using only labeled instances from the source domain. Although both these two methods are representation learning approaches, their objectives are quite different, and can be understood as complementary to each other.
In this paper we develop a unified network that is able to learn feature representations which are invariant to the shift between source and target domains, while at the same time being robust to reconstruction errors. The model is based on the recently proposed domain adversarial neural network (Ajakan et al., 2014; Ganin et al., 2016), where the goal is to learn representations that are informative for the desired learning task at the source domain, while at the same time being invariant to the shift between the source and target domains. The proposed model improves over the domain adversarial neural network by incorporating an auxiliary auto-encoder that works as an unsupervised regularizer to help the training of representations. Such a model naturally incorporates the above two complementary methods in a unified framework, while only incurs little computational overhead. In summary, our model is designed to achieve the following three objectives simultaneously in a unified framework: 1). It learns representations that are informative for the main learning task at the source domain. 2). It learns invariant features that are indistinguishable between the source and the target domains. 3). It is able to reconstruct the original input instances. To validate the effectiveness of the proposed model in domain adaptation, we compare it with state-of-the-art models in the literature on the Amazon benchmark data set for sentiment analysis. We show that our model consistently outperforms baseline methods while only incurring little computational overhead.

2 Related Work

There are two main streams of research for domain adaptation. The first one is based on reweighing scheme that aims to match the source and target domain distributions (Huang et al., 2006; Gong et al., 2013). The other line of works focus on learning feature transformations such that the feature distributions in the source and target domain are close to each other (Ben-David et al., 2007, 2010; Ajakan et al., 2014; Ganin et al., 2016). Both approaches have well-justified theoretical foundations (Huang et al., 2006; Ben-David et al., 2010) to guarantee their successes under proper assumptions, however in practice it was observed that unsupervised pretraining using stacked denoising auto-encoders (Vincent et al., 2008; Chen et al., 2012) often improves the generalization accuracy (Ganin et al., 2016).
The general approach for domain adaptation starts from algorithms that focus on linear hypothesis class (Blitzer et al., 2006; Germain et al., 2013; Cortes and Mohri, 2014). The linear assumption can be relaxed and extended into the non-linear setting using kernel trick, leading to a reweighting scheme that can be efficiently solved via quadratic programming (Huang et al., 2006; Gong et al., 2013). Recently, due to the availability of rich data and powerful computational resources, non-linear representations and hypothesis classes have been increasingly explored (Glorot et al., 2011; Baktashmotlagh et al., 2013; Chen et al., 2012; Ajakan et al., 2014; Ganin et al., 2016). This line of works focuses on building common and robust feature representations among multiple domains using either supervised neural networks (Glorot et al., 2011), or unsupervised pretraining using denoising auto-encoders (Vincent et al., 2008, 2010).
Adversarial training techniques which aim to build feature representations that are indistinguishable between source and target domains have been proposed in the last few years (Ajakan et al., 2014; Ganin et al., 2016). Specifically, one of the central ideas is to use neural networks, which are powerful function approximators, to approximate a distance measure known as $\mathcal{H}$-divergence between two domains (Kifer et al., 2004; Ben-David et al., 2007, 2010). The overall algorithm can be viewed as a zero-sum two-player game: one network tries to learn feature representations that can fool the other network, whose goal is to distinguish representations generated from the source domain between those generated from the target domain. The goal of the algorithm is to find a Nash-equilibrium of the game, or the stationary point of the min-max saddle point problem. Ideally, at such equilibrium state, feature representations from the source domain will share the same distributions as those from the target domain, and as a result, better generalization on the target domain can be expected by training models using only labeled instances from the source domain.
Theoretical work on domain adaptation is abundant. Kifer et al. (2004) proposed the $\mathcal{H}$-divergence to measure the similarity between two domains and derived a generalization bound on the target domain using empirical error on the source domain and the $\mathcal{H}$-divergence between the source and the target. This idea has later been extended to multi-source domain adaptation (Blitzer et al., 2008) and the corresponding generalization bound has been developed as well. We refer readers to Ben-David et al. (2010) for a thorough treatment on this subject. Concurrently, Mansour et al. (2009a) derived learning bounds on domain adaptation using discrepancy distance, which generalizes $\mathcal{H}$-divergence from binary loss function to arbitrary loss functions. See (Cortes et al., 2008; Mansour et al., 2009a,b,a) for more details.

3 A Theoretical Model for Domain Adaptation

In Sec. 1 we briefly mention that in order to hope for a successful domain adaptation algorithm, source domain and target domain should be similar to each other in a certain sense. Clearly we need to define a distance metric in order to measure the similarity of two distributions. Several notions of distance have been proposed for the purpose of domain adaptation (Ben-David et al., 2007, 2010; Mansour et al., 2009a,b). In this paper we will focus on using the $\mathcal{H}$-divergence (Ben-David et al., 2007, 2010), and the reasons for our choice will be clear in the sequel.

Definition 1 ((Ben-David et al., 2007, 2010))**.** Given two domain distributions $\mathcal{D}_S^X$ and $\mathcal{D}_T^X$ over X, and a hypothesis class $\mathcal{H}$, the $\mathcal{H}$-divergence between $\mathcal{D}_S^X$ and $\mathcal{D}_T^X$ is

$$d_{\mathcal{H}}(\mathcal{D}_S^X, \mathcal{D}_T^X) = 2 \sup_{\eta \in \mathcal{H}} \left| \Pr_{\mathbf{x} \sim \mathcal{D}_S^X}[\eta(\mathbf{x}) = 1] - \Pr_{\mathbf{x} \sim \mathcal{D}_T^X}[\eta(\mathbf{x}) = 1] \right| \tag{1}$$

One notable property of Def. 1 is that it depends on the capacity of the hypothesis class $\mathcal{H}$ to distinguish between the source and target distributions. More specifically, if we define

$$\mathcal{H}^{-1} = \{\eta^{-1}(\{1\}) \mid \eta \in \mathcal{H}\}$$

then we have $\mathcal{H}^{-1} \subseteq 2^X$, and $d_{\mathcal{H}}(\cdot, \cdot)$ will have the following equivalent definition:

$$d_{\mathcal{H}}(\mathcal{D}_S^X, \mathcal{D}_T^X) = 2 \sup_{A \in \mathcal{H}^{-1}} \left| \Pr_{\mathcal{D}_S^X}[A] - \Pr_{\mathcal{D}_T^X}[A] \right|$$

Readers who are familiar with the *total variation* distance, or the L^1 distance, may find the above definition closely related. In fact, when $\mathcal{H}$ contains all the possible measurable functions from X to $\{0, 1\}$, or equivalently, when $\mathcal{H}^{-1}$ contains all the measurable subsets of X, then our definition of $d_{\mathcal{H}}(\cdot, \cdot)$ reduces to the total variation distance. In this case, $d_{\mathcal{H}}(\mathcal{D}_S^X, \mathcal{D}_T^X) = 0$ iff $\Pr_{\mathcal{D}_S^X}(\cdot) = \Pr_{\mathcal{D}_T^X}(\cdot)$ almost surely. Consider another extreme case where $\mathcal{H}$ only contains two constant functions that either label all the instances to be 0 or to be 1, (equivalently, $\mathcal{H}^{-1} = \{\varnothing, X\}$), then $d_{\mathcal{H}}(\mathcal{D}_S^X, \mathcal{D}_T^X) = 0$ holds for every $\mathcal{D}_S^X, \mathcal{D}_T^X$. It is now clear that Def. 1 is flexible in the sense that it allows practitioners to define a hierarchy of distance functions with different granularties based on the complexity of our hypothesis space $\mathcal{H}$. As we will see shortly, this is also a necessary condition in order to derive a finite sample generalization bound based on VC theory.
Assume that $\mathcal{H}^{-1}$ is symmetric in the following sense: if $C \in \mathcal{H}^{-1}$, we also have $X \backslash C \in \mathcal{H}^{-1}$, then it can be shown that the following equality holds:

$$d_{\mathcal{H}}(\mathcal{D}_S^X, \mathcal{D}_T^X) = 2 \left(1 - \inf_{\eta \in \mathcal{H}} \left(\Pr_{\mathbf{x} \sim \mathcal{D}_S^X}[\eta(\mathbf{x}) = 0] + \Pr_{\mathbf{x} \sim \mathcal{D}_T^X}[\eta(\mathbf{x}) = 1] \right) \right) \tag{2}$$

In practice since we do not have access to the true data generation distributions $\mathcal{D}_S^X$ and $\mathcal{D}_T^X$, we cannot hope to be able to compute Eq. 2 exactly. However, using unlabeled samples from both the source and the target domain, we can obtain an estimator of the $\mathcal{H}$-divergence as:

$$\hat{d}_{\mathcal{H}}(S,T) = 2\left(1 - 2\min_{\eta\in\mathcal{H}}\left[\frac{1}{2n}\sum_{i=1}^{n}\mathbb{I}[\eta(\mathbf{x}_i)=0] + \frac{1}{2n}\sum_{i=n+1}^{2n}\mathbb{I}[\eta(\mathbf{x}_i)=1]\right]\right) \tag{3}$$

where the first n samples come from the source domain and last n samples are from the target domain. $\mathbb{I}[\eta(\mathbf{x})=1]$ takes value 1 iff $\mathbf{x}$ is a sample from the source domain. Hence the term inside the min function in the above function can be understood as the minimum classification error in discriminating instances from the source and the target domains.
For any interesting hypothesis class $\mathcal{H}$, the empirical $\mathcal{H}$-divergence can also be hard to compute exactly due to the minimization over the binary classification error, which is NP-hard even for simple hypothesis class like half-spaces (Guruswami and Raghavendra, 2009). In practice practitioners resort to convex relaxation techniques to smooth the binary classification loss function into other loss functions, e.g., hinge loss, log-loss, etc. With the help of the empirical $\mathcal{H}$-divergence, Ben-David et al. (2007) proves the following generalization bound for domain adaptation:

Theorem 1 (Ben-David et al. (2007))**.** Let $\mathcal{H}$ be a hypothesis class of VC-dimension d. With probability $1-\delta$ over the choice of samples $S\sim(\mathcal{D}_S^X)^n$ and $T\sim(\mathcal{D}_T^X)^n$, for every $\eta\in\mathcal{H}$:

$$R_{\mathcal{D}_T}(\eta) \le R_S(\eta) + \sqrt{\frac{4}{n}(d\log\frac{2en}{d}+\log\frac{4}{\delta})} + 4\sqrt{\frac{1}{n}(d\log\frac{2n}{d}+\log\frac{4}{\delta})} + \hat{d}_{\mathcal{H}\Delta\mathcal{H}}(S,T) + \beta$$

with $\beta \ge \inf_{\eta^*\in\mathcal{H}}[R_{\mathcal{D}_S}(\eta^*) + R_{\mathcal{D}_T}(\eta^*)]$, $R_S(\eta) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}[\eta(\mathbf{x}_i)\ne y_i]$ is the empirical source risk, and $\mathcal{H}\Delta\mathcal{H}$ is the symmetric difference of $\mathcal{H}$.

Intuitively, the theorem above means that, under the setting of domain adaptation, the generalization error can be low only if the following conditions hold:

1. β is small, i.e., there exists a hypothesis that is able to achieve a low risk on both the source and the target distributions.[1]
2. The learning algorithm is able to find representations under which the empirical $\mathcal{H}$-divergence $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(S,T)$ between samples from S and T is small.
3. The hypothesis class should be rich enough so that the empirical error on the source domain $R_S(\eta)$ is small.

Note that the generalization bound also introduces a natural tradeoff in choosing the complexity of the hypothesis class $\mathcal{H}$. Based on Thm. 1, Ajakan et al. (2014); Ganin and Lempitsky (2015) designed a neural network model that aims to learn representations which are both discriminative for the main learning task, i.e., to minimize $R_S(\eta)$, while at the same time are indistinguishable between the source and the target domains, i.e., to minimize $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(S,T)$. Also note that the min part of $\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(S,T)$ should be interpreted as the error rate of the domain classifier. A small domain classification error implies to a large distance function between S and T. On the other hand, if the error rate is large, then S and T are close to each other.

[1]It is certainly possible that there is a hypothesis that achieves low risk on the target domain and a high risk on the source domain. But we cannot hope to find such hypothesis if we only have access to labeled instances from the source domain.

4 Method

In this section we shall first describe the domain adversarial neural network (Ajakan et al., 2014; Ganin and Lempitsky, 2015; Ganin et al., 2016) (DANN), and then propose our model, which we term *domain adversarial auto-encoder (DAuto)*, based on DANN.

4.1 Domain Adversarial Neural Network

The domain adversarial neural network (DANN) model proposed by Ajakan et al. (2014); Ganin and Lempitsky (2015) is shown as follows:

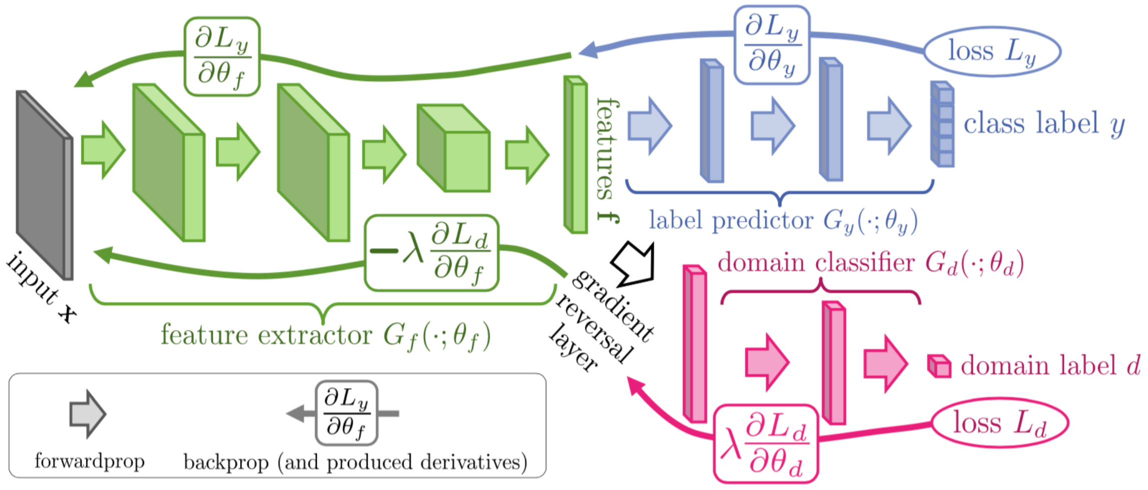
$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
features $\mathbf{f}$
class label y
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
input x
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
domain label d
$\lambda\frac{\partial L_d}{\partial \theta_d}$
loss L_d
forwardprop
backprop (and produced derivatives)

Figure 1: Figure courtesy of Ganin and Lempitsky (2015). DANN contains three parts. The feature learning part (in green) with parameter θ_f, the main classification part (in blue) with parameter θ_y and the domain classification part (in red) with parameter θ_d. The gradients from the main classification part and the domain classification part are combined together to update the parameters of the feature learning component.

Let $\mathcal{L}_y(\mathbf{x}, y; \theta_f, \theta_y)$ denote the main classification loss function and $\mathcal{L}_d(\mathbf{x}; \theta_f, \theta_d)$ denote the surrogate loss of the binary loss function for the domain classifier. In view of Thm. 1, once we fix our model and the source and target domain data sets, we can only minimize the generalization bound by minimizing the first and fourth terms in the upper bound, leading to the following optimization problem:

$$\min \quad R_S(\eta) + \lambda \cdot \hat{d}_{\mathcal{H}\Delta\mathcal{H}}(S, T)$$

where $\lambda > 0$ is a hyperparameter to tradeoff the relative importances of these two terms. Plug in the model shown in Fig. 1 and the defined loss function Eq. 3, ignoring all the constant terms, we can rewrite the objective function as

$$\begin{aligned}
&\min \quad R_S(\eta) + \lambda \cdot \hat{d}_{\mathcal{H}}(S,T) \\
&= \min_{\theta_f,\theta_y} \quad \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_y(\mathbf{x}_i, y_i; \theta_f, \theta_y) - \lambda \cdot \min_{\theta_d} \left(\frac{1}{2n}\sum_{i=1}^{n} \mathcal{L}_d(\mathbf{x}_i; \theta_f, \theta_d) + \frac{1}{2n}\sum_{i=n+1}^{2n} \mathcal{L}_d(\mathbf{x}_i; \theta_f, \theta_d) \right) \\
&= \min_{\theta_f,\theta_y} \max_{\theta_d} \quad \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_y(\mathbf{x}_i, y_i; \theta_f, \theta_y) - \lambda \left(\frac{1}{2n}\sum_{i=1}^{n} \mathcal{L}_d(\mathbf{x}_i; \theta_f, \theta_d) + \frac{1}{2n}\sum_{i=n+1}^{2n} \mathcal{L}_d(\mathbf{x}_i; \theta_f, \theta_d) \right) \qquad (4)
\end{aligned}$$

(4) is a minimax saddle point optimization problem. Due to the use of nonlinear activation function, each sub-problem is still nonconvex even when all the other parameters are fixed in (4). This property makes (4) hard to optimize in practice. The min-max optimization formulation above is very similar to the recently proposed generative adversarial network (Goodfellow et al., 2014), whose primary goal is to generate samples that are hard to distinguish from the true instances in the training data set. Both these two problems can be interpreted as a zero-sum game between two players due to their minimax formulation.

A principled way to optimize the above min-max objective function is to employ an alternative optimization method where for each fixed pair $(\hat{\theta}_f, \hat{\theta}_y)$ the algorithm optimizes over θ_d until convergence and then with fixed $\hat{\theta}_d$ the algorithm optimizes the pair (θ_f, θ_y) until convergence. In practice the authors adopt a stochastic approach where at each iteration they sample a mini-batch instance pairs from the source domain and a mini-batch sample of instances from the target domain. Then the gradients can be computed correspondingly by computing the objective function using the current iterate of parameters. Note that there is an caveat that in order to correctly implement the optimization, the partial derivative $\partial\mathcal{L}_d(\mathbf{x}; \theta_f, \theta_d)/\partial\theta_d$ should be reversed when propagated to θ_f. Such reversal operation is implemented as a gradient reversal layer (GRL) as shown in Fig. 1. In the forward phase the GRL simply works as an identity operator while in the backward phase the GRL layer inverts the gradient that flow through it by -1.

4.2 Domain Adversarial Auto-encoder

4.2.1 Motivation

DANN is easy to interpret and is well-justified by minimizing two terms in the generalization upper bound shown in Thm. 1. On the other hand, marginalized stacked denoising auto-encoder (Chen et al., 2012, mSDA) has been widely validated to be helpful and efficient in domain adaptation by unsupervised pretraining. It has also been observed experimentally that the performance of DANN gets further improved when trained with representation learned by mSDA (Ganin et al., 2016). Both DANN and mSDA are representation learning methods, but their objective functions are quite different. Hence a natural question to ask is: can we design a unified model that incorporates the advantages of both models simultaneously, instead of building a pipeline system that first does unsupervised pretraining and then learns invariant representation?

In this section we shall mainly discuss a unified model such that: 1). It learns representations that are informative for the main learning task at the source domain. 2). It learns invariant features that

are indistinguishable between the source and the target domains. 3). It is able to reconstruct the original input instances. We term the proposed model *DAuto*. The first requirement corresponds to a term in the objective function that aims to minimize the empirical classification error at the source domain. To realize the second goal, we take the same idea as DANN and uses a discriminative classifier to approximate the minimum domain classification error. Forour third design goal, we use a stacked auto-encoder to enforce the representations to be robust under injected noise. This idea is similar to the recent Ladder network (Rasmus et al., 2015) for semi-supervised learning, where authors augment a supervised learning task with a hierarchy of unsupervised denoising auto-encoders.
DAuto is based on the above DANN architecture. We augment the DANN model by viewing the representation learning part of DANN, i.e., the part with parameters θ_f, as an encoding process. Then it is quite a natural idea to construct a corresponding decoding process, from which we can further regularize our objective function to achieve our third goal: the learned representation should be able to reconstruct the original input. The idea of using auto-encoder (Vincent et al., 2010) and its variants for unsupervised pretraining is well-known (Erhan et al., 2010; Bengio et al., 2013). Feature learned from these auto-encoders is usually robust to noise and has been validated experimentally to be helpful for clustering, etc. However, instead of first training an mSDA to learn the desired feature representation and then feed it into the DANN, we design our network structure so that it contain the reconstruction loss as one part of its objective function, which effectively combines the auto-encoder as one component of the overall network structure.

4.2.2 Model Description

We first show the model architecture of DAuto in Fig. 2. DAuto contains four major components in its design: the feature learning part (in green) with parameter θ_f, the main classification part (in blue) with parameter θ_y, the domain classification part (in red) with parameter θ_d and the unsupervised auto-encoder part (in purple) with parameter θ_r. The feature learning component provides a shared representation for all the other parts of the model, and as a result the gradients from the main classification part, the domain classification part and the auto-encoder part are combined together to update the parameters of the feature learning part.
More specifically, let $G_f(\cdot;\theta_f)$ be the feature learning component that maps an input instance from $\mathbb{R}^d$ to a hidden feature representation $\mathbb{R}^D$, i.e., $G_f(\mathbf{x};\theta_f) \in \mathbb{R}^D$. Similarly, let $G_y(\cdot;\theta_y)$ be the main classification part, which is a function from $\mathbb{R}^D$ to the output space, e.g., Δ^k when the prediction contains k classes. $G_d(\cdot;\theta_d)$ now corresponds to the domain classification part of DAuto, i.e., a function from $\mathbb{R}^D$ to $[0,1]$, where the output of $G_d(\cdot;\theta_d)$ measures how confident the domain classifier is about the event that the input instance comes from the target domain. Finally, we use $G_r(\cdot;\theta_r) : \mathbb{R}^D \to \mathbb{R}^d$ to represent the auto-encoder component (more precisely, $G_r(\cdot;\theta_r)$ corresponds to the decoding function while $G_f(\cdot;\theta_f)$ is treated as the encoding function). As shown as the purple part in Fig. 2, $G_r(\cdot;\theta_r)$ is a "mirror" of $G_f(\cdot;\theta_f)$: for example, suppose the representation learning part contains 3 fully-connected layers with number of units $100, 200$ and 300 in the corresponding layers. Then the decoding part will contain exactly three layers with $300, 200$ and 100 units, respectively. Such design allows us to incorporate the following reconstruction loss

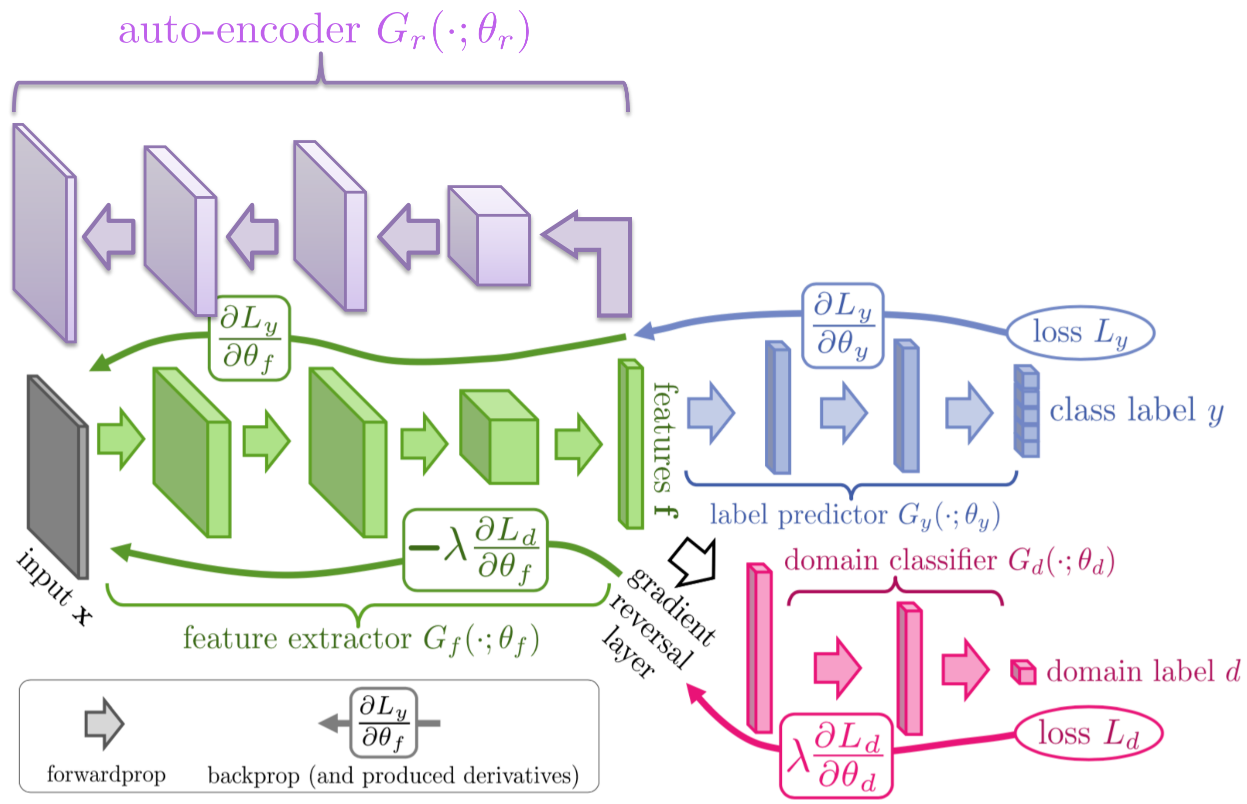
auto-encoder $G_r(\cdot;\theta_r)$
$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
features $\mathbf{f}$
class label y
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
input $\mathbf{x}$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
feature extractor $G_f(\cdot;\theta_f)$
domain label d
loss L_d
$\lambda\frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

Figure 2: Figure modified from Ganin et al. (2016). Model architecture of domain adversarial auto-encoder (DAuto). DAuto contains four parts. The feature learning part (in green) with parameter θ_f, the main classification part (in blue) with parameter θ_y, the domain classification part (in red) with parameter θ_d and the unsupervised auto-encoder part (in purple) with parameter θ_r. The gradients from the main classification part, the domain classification part and the auto-encoder part are combined together to update the parameters of the feature learning part.

as a regularizer into the DAuto model:

$$\mathcal{L}_r = \frac{1}{2n}\sum_{i=1}^{n}\mathcal{L}_r(\mathbf{x}_i;\theta_f,\theta_r) + \frac{1}{2n}\sum_{i=n+1}^{2n}\mathcal{L}_r(\mathbf{x}_i;\theta_f,\theta_r)$$

where we recall the first n samples come from the source domain and the last n samples come from the target domain. Take the model architecture shown in Fig. 2 as an example. Let $\mathbf{x}$, $\mathbf{f}^{(1)}$, $\mathbf{f}^{(2)}$, $\mathbf{f}^{(3)}$ and $\mathbf{f}$ be the input instance, the hidden vector at the first, second, third and last layer of $G_f(\cdot;\theta_f)$. Correspondingly, let $\hat{\mathbf{f}}^{(3)}$, $\hat{\mathbf{f}}^{(2)}$, $\hat{\mathbf{f}}^{(1)}$ and $\hat{\mathbf{x}}$ be the reconstructed hidden features and input instance in the decoding part $G_r(\cdot;\theta_r)$, then we can build the following reconstruction loss $\mathcal{L}_r(\mathbf{x};\theta_f,\theta_r)$:

$$\mathcal{L}_r(\mathbf{x};\theta_f,\theta_r) = ||\hat{\mathbf{x}}-\mathbf{x}||_2^2 + ||\hat{\mathbf{f}}^{(1)}-\mathbf{f}^{(1)}||_2^2 + ||\hat{\mathbf{f}}^{(2)}-\mathbf{f}^{(2)}||_2^2 + ||\hat{\mathbf{f}}^{(3)}-\mathbf{f}^{(3)}||_2^2 \tag{5}$$

In this example we use the squared ℓ_2 distance only for illustration, other distance metrics can be used as well. Note that $\mathcal{L}_r$ also depends on θ_f because all the reconstructed hidden features follow from the last layer in the encoder. Combining the reconstruction loss into the objective function, we reach

$$\begin{aligned}\min_{\theta_f,\theta_y,\theta_r}\max_{\theta_d}\quad & \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y(\mathbf{x}_i,y_i;\theta_f,\theta_y)-\lambda\left(\frac{1}{2n}\sum_{i=1}^{n}\mathcal{L}_d(\mathbf{x}_i;\theta_f,\theta_d)+\frac{1}{2n}\sum_{i=n+1}^{2n}\mathcal{L}_d(\mathbf{x}_i;\theta_f,\theta_d)\right)\\ & +\mu\left(\frac{1}{2n}\sum_{i=1}^{n}\mathcal{L}_r(\mathbf{x}_i;\theta_f,\theta_r)+\frac{1}{2n}\sum_{i=n+1}^{2n}\mathcal{L}_r(\mathbf{x}_i;\theta_f,\theta_r)\right)\end{aligned} \tag{6}$$

Again, we emphasize that the first n labeled samples are from the source domain and the last n unlabeled samples are from the target domain. The above objective function can be optimized using existing adaptive gradient method like AdaGrad, etc. Note that both the second and the third terms in (6) can be understood as regularizers that help to learn invariant and robust feature representations simultaneously. Along with the first term in (6), the objective function seeks to learn representations that on one hand are informative for our main learning task, while at the same time are robust and invariant between source and target domains.

4.2.3 Auto-encoder as Marginal Probability

One may ask why adding a reconstruction loss as regularizers into the existing DANN objective will help domain adaptation? In this section we provide a probabilistic justification showing that along with the cross-entropy loss, the reconstruction loss can be interpreted as maximizing a joint probability distribution over both instances and their labels under proper modeling assumptions.
Let $\hat{y}_i^{(k)}$ be the k-th output of the softmax classifier with input instance $\mathbf{x}_i$, i.e., $\hat{y}_i^{(k)} = \Pr(y_i = k \mid \mathbf{x}_i;\theta_f,\theta_y)$. It is clear that the usual cross-entropy loss corresponds to the negative of the log conditional likelihood function when the true generating distribution is approximated by the empirical distribution on data set: $-\sum_k \mathbb{I}_{y_i=k}\log\hat{y}_i^{(k)} = -\log\Pr(y_i = k_i \mid \mathbf{x}_i;\theta_f,\theta_y)$, where k_i is the label of $\mathbf{x}_i$. The neural network parametrized with θ_f and θ_y then gives a conditional distribution $y \mid \mathbf{x}$. Now from the Bayesian perspective, consider the joint distribution over all the variables, i.e., $\mathbf{x}$, y and all the model parameters. The joint distribution of this model can be described as:

$$p(\mathbf{x},y,\theta_f,\theta_y,\hat{\theta}_f,\theta_r) = p(\mathbf{x};\hat{\theta}_f,\theta_r)p(y\mid\mathbf{x};\theta_f,\theta_y)p(\theta_f,\theta_y,\hat{\theta}_f,\theta_r) \tag{7}$$

where $p(\theta_f,\theta_y,\hat{\theta}_f,\theta_r)$ is the prior distribution over model parameters. We shall show that by choosing a proper form of the prior distribution as well as the marginal distribution $p(\mathbf{x};\hat{\theta}_f,\theta_r)$, the objective function (6) of our model corresponds to the negative log joint likelihood function augmented with the domain adversarial regularizer.
Given a set of training instances $\{\mathbf{x}_i\}_{i=1}^n$, consider the kernel density estimation parametrized by two neural networks $G_f(\cdot;\hat{\theta}_f)$ and $G_r(\cdot;\theta_r)$:

$$p(\mathbf{x};\hat{\theta}_f,\theta_r) \propto \frac{1}{nh}\sum_{i=1}^{n}K\left(\frac{\mathbf{x}-G_r(G_f(\mathbf{x}_i;\hat{\theta}_f);\theta_r)}{h}\right) \tag{8}$$

where $h > 0$ is the bandwidth and $K(\cdot)$ is the kernel function. Now consider evaluating the density function (8) on instance $\mathbf{x}_j$ from the training set:

$$\begin{aligned}\log p(\mathbf{x}_j; \hat{\theta}_f, \theta_r) &= \log\left(\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}_j - G_r(G_f(\mathbf{x}_i; \hat{\theta}_f); \theta_r)}{h}\right)\right) \\ &\geq \log K\left(\frac{\mathbf{x}_j - G_r(G_f(\mathbf{x}_j; \hat{\theta}_f); \theta_r)}{h}\right) - \log(nh) \qquad (9)\end{aligned}$$

If we choose $K(\cdot)$ to be a specific form of kernel function, e.g., a Gaussian kernel, then (9) can be simplified as follows:

$$\log p(\mathbf{x}_j; \hat{\theta}_f, \theta_r) \propto -\mu \cdot ||\mathbf{x}_j - G_r(G_f(\mathbf{x}_j; \hat{\theta}_f); \theta_r)||_2^2$$

where $\mu > 0$ is a constant term that depends on the band-width h but not $\mathbf{x}_j$. Note that other choices of kernel functions can lead to different kinds of reconstruction loss, e.g., a Laplace kernel leads to an ℓ_1 measure of the reconstruction loss. The above derivation shows that the reconstruction loss of an auto-encoder corresponds to a lower bound on the marginal log probability $\log p(\mathbf{x}; \hat{\theta}_f, \theta_r)$ where $p(\cdot)$ is given by a kernel density estimation. It is worth pointing out that the lower bound in (9) becomes more and more accurate as $h \to 0$ or $G_r \circ G_f$ becomes close to an identity function, which is exactly the design purpose of an auto-encoder.
As a result, if we maximize the log of the joint likelihood function $\log p(\mathbf{x}; \hat{\theta}_f, \theta_r) + \log p(y \mid \mathbf{x}; \theta_f, \theta_y)$ over labeled instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ from the source domain and unlabeled instances $\{\mathbf{x}_i\}_{i=n+1}^{2n}$ from the target domain, we obtain:

$$\min_{\hat{\theta}_f, \theta_r, \theta_f, \theta_y} \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_y(\mathbf{x}_i, y_i; \theta_f, \theta_y) + \mu\left(\frac{1}{2n}\sum_{i=1}^{n} \mathcal{L}_r(\mathbf{x}_i; \hat{\theta}_f, \theta_r) + \frac{1}{2n}\sum_{i=n+1}^{2n} \mathcal{L}_r(\mathbf{x}_i; \hat{\theta}_f, \theta_r)\right) \qquad (10)$$

The above interpretation not only explains the empirical success of auto-encoders as regularizers in discriminative tasks, but also provides us a way to take advantage of unlabeled instances in a principled way. Our derivation also implies the possibility of applying auto-encoder based approaches in semi-supervised learning, which has recently been explored by Rasmus et al. (2015) in their Ladder networks.
From Fig. 2, the model parameter $\hat{\theta}_f$ in the generation process of $\mathbf{x}$ and the parameter θ_f in discriminative function $p(y \mid \mathbf{x})$ are shared. We can enforce this constraint by specifying our prior distribution over model parameters as follows:

$$p(\hat{\theta}_f, \theta_r, \theta_f, \theta_y) \propto p_0(\hat{\theta}_f, \theta_r, \theta_f, \theta_y) \cdot \delta(\hat{\theta}_f - \theta_f)$$

where p_0 is a flat prior and $\delta(\cdot)$ is the Dirac delta function. Incorporating this prior into (10), we can see that (10) is exactly (6) without the domain adversarial term. As a result, the objective function in (6) can be understood as an implementation of the maximum-a-posteriori (MAP) estimation of model parameters $\{\theta_f, \theta_r, \theta_y\}$ augmented with the domain adversarial regularizer.

5 Experiments

5.1 Experimental Setting

In this section, we conduct experiments to compare DAuto with a variety of methods for domain adaptation on the Amazon data set. The Amazon data set consists of reviews of products on Amazon (Blitzer et al., 2007). The task is to predict the polarity of a text review, i.e., whether the review for a specific product is positive or negative. The data set contains text reviews for the following four categories: books, DVDs, electronics and kitchen appliances. Each of the product contains 2000 text reviews as training data, and 3000∼5000 reviews as test data. Each text review is described by a feature vector of 5000 dimensions, where each dimension correspond to a word in the dictionary. The data set is a benchmark data that has been frequently used for the purpose of sentiment analysis (Blitzer et al., 2006, 2007; Chen et al., 2012; Ajakan et al., 2014; Ganin and Lempitsky, 2015), and is publicly available at `http://www.cs.jhu.edu/~mdredze/datasets/sentiment/`. We list the detailed statistics of the Amazon data set in Table. 1.

Table 1: Statistics about the Amazon data set.

Data set	**Train**	**Test**	**Feature**
Books (B)	2000	4465	5000
DVD (D)	2000	3586	5000
Electronics (E)	2000	5681	5000
Kitchen (K)	2000	5945	5000

In view of the four categories in the Amazon data set, there are 16 possible binary classification tasks, i.e., $\langle S, T\rangle$, where $S, T \in \{\text{B, D, E, K}\}$. Among the 16 possible tasks, 12 of them are valid domain adaptation settings, while the rest 4 experiments form standard classification setting where the training and test distributions are approximately the same. Nevertheless, we shall still compare all the methods described below on all the 16 experiments, mainly to check that a successful domain adaptation algorithm should be adaptative in the sense when there is no shift between the source and the target domain, the algorithm should still be able to achieve a good generalization error. Hence for each source-target pair, we train the corresponding models completely on labeled source instances with unlabeled target instances. Since each task is a binary classification problem, we use the classification accuracy on the target domain as our main metric to evaluate the performance of different models.

We compare DAuto with the following methods:

1. **Multilayer Perceptron (MLP)**. This is the baseline model which ignores the possible shifts between source and target domains. During training process MLP does not need to have access to the unlabeled instances from target domains. Throughout all the experiments, we fix the structure of MLP to have one hidden layer with 500 hidden units. We use `AdaDelta` (Zeiler, 2012) to train the MLP with a dropout rate (Srivastava et al., 2014) 0.01. We also use weight decay with coefficient 0.01. The activation function in the hidden layer is the rectified linear unit (Nair and Hinton, 2010).

2. **mSDA**. mSDA pretrains all the unlabeled instances from both the source and the target domains to build a feature map for the input space. The constructed representations from mSDA are used to train a linear SVM classifier as suggestd in the original paper (Chen et al., 2012). In all the experiments, we set the corruption level to be 0.5 in training mSDA, and stack 1 layers of denoising auto-encoders. While being a simple model, mSDA forms a very strong baseline on the Amazon data set that is hard to beat (Ajakan et al., 2014).

3. **Ladder Network (Ladder)**. The Ladder network (Rasmus et al., 2015) is a novel structure aiming for semi-supervised learning. It is a hierarchical denoising auto-encoder where reconstruction errors between each pair of hidden layers are incorporated into the objective function. The Ladder network can have access to the unlabeled instances from target domains as a means to utilize the unlabeled data in a unsupervised way. We implement the Ladder network as described in the original paper (Rasmus et al., 2015), but again only use one hidden layer with 500 units. The other experimental setting for Ladder are set to the same as MLP, i.e., the same activation function, the same dropout rate, etc.

4. **Domain Adversarial Neural Network (DANN)**. DANN is a recent model proposed for domain adaptation using adversarial training techniques. We detailed its discussion in Sec. 3. Again, we use a one hidden layer fully-connected network with 500 units as the main classifier, and we use logistic regression with the hidden layer as input to build the domain classifier. To approximate the binary loss, we use the cross-entropy loss function as the error function of the domain classifier. We set $\lambda = 0.001$ as the tradeoff parameter between the main classification loss and the domain classification loss.

For our proposed model DAuto, again, in order to make a fair comparison with MLP, Ladder and DANN, we use an MLP with only one hidden layer (500 units) as the main prediction component. As in the DANN model, we use a logistic regression to implement the domain classification part. We use stacked auto-encoder (without injected noise) to implement the decoding part: the decoding model is also an MLP with one hidden layer with exactly the same number of units. But we do not share the weights of the decoding MLP with the encoding one, for the following reason: the encoder in DAuto is also expected to learn invariant features between these two domains while the decoder is only responsible for the reconstruction error. In all the 16 experiments, we fix $\lambda = \mu = 0.001$ as the tradeoff parameters. Again, we use `AdaDelta` with mini-batches to train the model. Dropout rate is set to be 0.7.
We implement all the models ourselves in Python. For neural network based models, we use Theano (Bergstra et al., 2010) with Keras library to implement them. Whenever possible, we run our models on GPUs.

5.2 Results and Analysis

We report the classification accuracies on the test data of the 16 pairs of tasks to have a thorough comparisons among the 5 models in Table. 2. DAuto outperforms all the other baseline methods on 14 out of 16 tasks, whereas DANN achieves the best test set accuracy on $D \to E$ and $D \to K$. To check whether the accuracy differences between those five methods are significant or not, we perform a paired t-test and report the two-sided p-value under the null hypothesis that two related paired samples have identical average values. We show the p-value matrix in Table. 3 where for each pair of methods, we report the maximum p-value among the 16 tasks. We note that DAuto performs consistently better than all the other competitors. Note that the only difference between DANN and DAuto is the auto-encoder regularizer that forces the feature learning part to learn robust features, we conclude that DAuto successfully helps to build robust representations.

Table 2: Binary classification accuracies on 16 tasks of 5 models: MLP, Ladder, mSDA, DANN and DAuto (ours). $S \to T$ means that S is the source domain and T is the target domain.

Task	**MLP**	**mSDA**	**Ladder**	**DANN**	**DAuto**
B→B	0.823	0.812	0.817	0.828	**0.834**
B→D	0.770	**0.785**	0.764	0.773	0.776
B→E	0.728	0.734	0.735	0.734	**0.749**
B→K	0.762	0.770	0.775	0.769	**0.782**
D→B	0.768	0.769	0.757	0.765	**0.776**
D→D	0.830	0.820	0.832	**0.834**	**0.834**
D→E	0.753	0.768	**0.784**	0.776	**0.784**
D→K	0.776	0.793	**0.802**	0.789	0.795
E→B	0.693	**0.726**	0.683	0.693	0.707
E→D	0.696	**0.741**	0.715	0.694	0.724
E→E	0.850	0.857	0.854	0.847	**0.864**
E→K	0.845	0.858	0.848	0.843	**0.863**
K→B	0.687	0.715	0.694	0.710	**0.723**
K→D	0.711	0.738	0.731	0.727	**0.748**
K→E	0.838	0.825	0.843	0.838	**0.849**
K→K	0.869	0.867	0.874	0.873	**0.882**

Table 3: p-value matrix between different domain adaptation algorithms. Each entry in the matrix corresponds to the maximum p-value under a paired t-test on 16 domain adaptation tasks from the Amazon data set.

	MLP	**mSDA**	**Ladder**	**DANN**	**DAuto**
MLP	-	5.75e-277	1.99e-49	1.19e-25	1.39e-33
mSDA	5.75e-277	-	6.99e-274	6.99e-274	1.30e-265
Ladder	1.99e-49	6.99e-274	-	1.31e-53	1.62e-51
DANN	1.19e-25	6.99e-274	1.31e-53	-	3.27e-32
DAuto	1.39e-33	1.30e-265	1.62e-51	3.27e-32	-

To show the effectiveness of different domain adaptation algorithms when labeled instances are scarce, we test the five algorithms on the 16 tasks by gradually increasing the size of the labeled training instances, but still use the whole test data set to measure the performance. More specifically, we use 0.2, 0.5, 0.8 and 1.0 fraction of the available labeled instances from source domain during training. A successful domain adaptation algorithm should be able to take advantage of

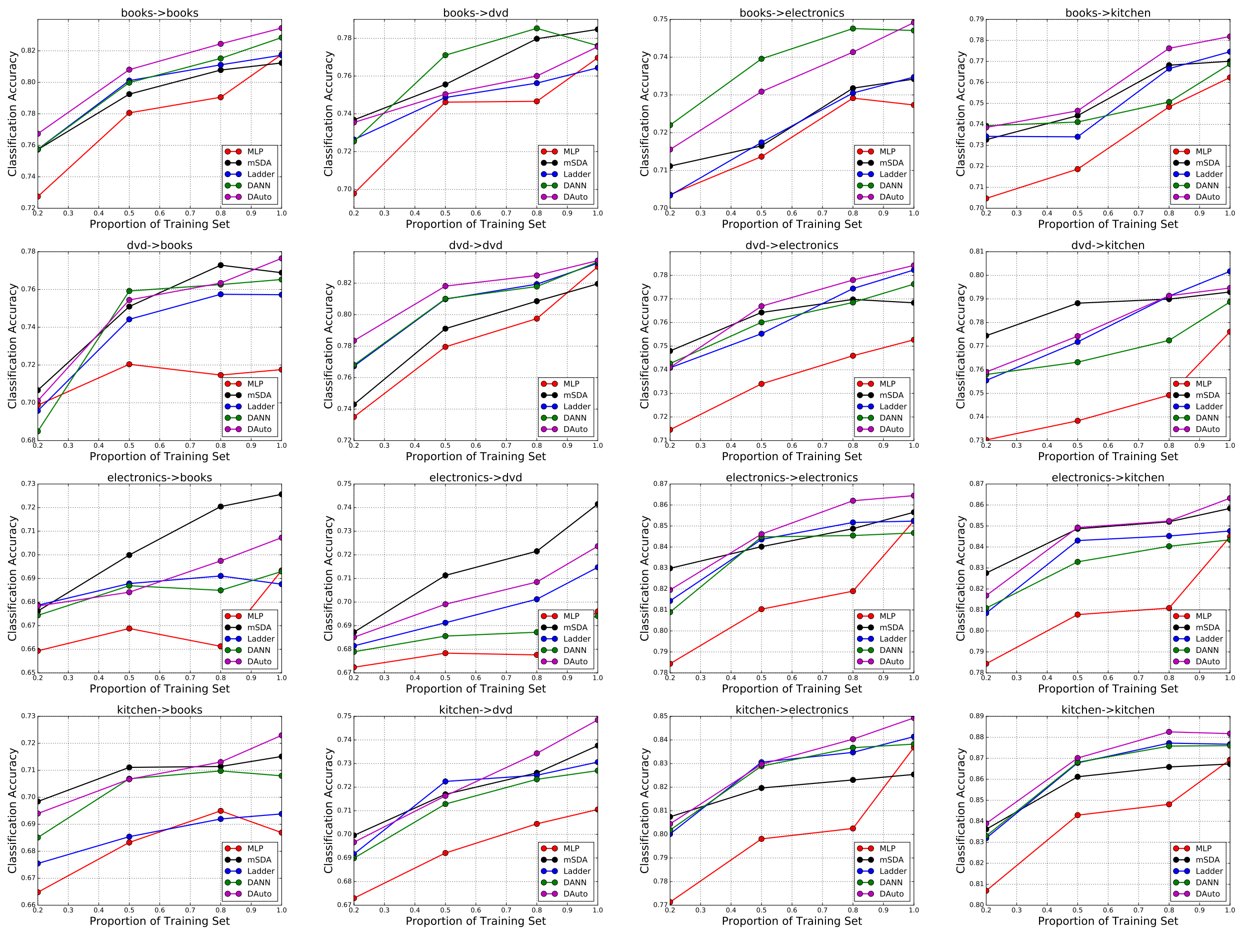

Figure 3: Test set performances of MLP, Ladder, mSDA, DANN and DAuto when the size of training set is gradually increasing from 0.2 to 1.0.

the unlabeled instances from the target domain to help generalization even when the amount of labeled instances available is small. We plot the results in Fig. 3. As can be seen from Fig. 3, all the methods, except Ladder, are able to utilize the unlabeled instance from the target domain to help generalization, when compared with the baseline MLP model. Ladder was originally proposed for semi-supervised learning, which assumes that the source and the target domains share the same distribution. This assumption might explain why Ladder does not perform as well as other domain adaptation methods.

6 Conclusion

We propose a unified network that is able to learn feature representations that are informative and robust, while at the same time being invariant between source and target domains. Our model is motivated by the recent advances in domain adaptation using representation learning approaches, namely the domain adversarial neural network and stacked denosing auto-encoder. Our model improves over these two models by incorporating both them into a unified framework. Such design only incurs little computational overhead while being able to take advantage of these two complementary approaches. To demonstrate the effectiveness of our model, we conduct detailed experiments on the Amazon benchmark data set, showing that our model consistently outperforms the state-of-the-art methods and other competitors as well. It is also worth pointing out that the underlying idea can also be seamlessly extended to dynamic domain as well, where we see various applications with time series data.

7 Acknowledgement

HZ gratefully acknowledges support from ONR contract N000141512365.

References

Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., and Marchand, M. (2014). Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*.

Baktashmotlagh, M., Harandi, M. T., Lovell, B. C., and Salzmann, M. (2013). Unsupervised domain adaptation by domain invariant projection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 769–776.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., et al. (2007). Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137.

Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013). Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: A cpu and gpu math compiler in python.

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2008). Learning bounds for domain adaptation. In *Advances in neural information processing systems*, pages 129–136.

Blitzer, J., Dredze, M., Pereira, F., et al. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

Chen, M., Xu, Z., Weinberger, K., and Sha, F. (2012). Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*.

Cortes, C. and Mohri, M. (2014). Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126.

Cortes, C., Mohri, M., Riley, M., and Rostamizadeh, A. (2008). Sample selection bias correction theory. In *International Conference on Algorithmic Learning Theory*, pages 38–53. Springer.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1180–1189.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Germain, P., Habrard, A., Laviolette, F., and Morvant, E. (2013). A pac-bayesian approach for domain adaptation with specialization to linear classifiers. In *ICML (3)*, pages 738–746.

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

Gong, B., Grauman, K., and Sha, F. (2013). Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *ICML (1)*, pages 222–230.

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

Guruswami, V. and Raghavendra, P. (2009). Hardness of learning halfspaces with noise. *SIAM Journal on Computing*, 39(2):742–765.

Huang, J., Gretton, A., Borgwardt, K. M., Schölkopf, B., and Smola, A. J. (2006). Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608.

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009a). Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009b). Multiple source adaptation and the rényi divergence. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 367–374. AUAI Press.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

Vapnik, V. N. and Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.