A Survey on Homogeneous Transfer Learning

Connor McIntee

May 5, 2023

1 Introduction

In machine learning, it's often assumed that the training and testing data are drawn independently and from the same distribution (*i.i.d.*). However, this assumption can be harmful to the model's efficacy when the training and test data are drawn from different distributions. The field of transfer learning tries to solve this problem by transferring knowledge from the training data to the test data. While transfer learning encompasses all shifts of the data Domains, including a differing set of training and testing features, we will focus on homogeneous transfer learning, where the shifts only occur in the probability distributions of the data.

Transfer learning is a valuable technique in machine learning, particularly in scenarios where finding suitable training data is difficult or expensive. By training models on data from related fields, transfer learning can enable the effective completion of different tasks. With the exponential growth of available data, transfer learning has become even more invaluable. By utilizing a wide range of data, transfer learning enables models to learn from a vast amount of information, even when the data may not be perfectly aligned with the problem at hand. Moreover, transfer learning can reduce the need for large amounts of labeled data, making it more cost-effective and efficient in many scenarios. As Big Data continues to grow, transfer learning has the potential to unlock new possibilities for data-driven applications and revolutionize the field of machine learning.

2 Problem and Notation

Before diving into the various applications and techniques in transfer learning, we first need to review the common notation used in the field. A Domain, $D = \{\mathcal{X}, P(X)\}$, consists of a feature space $\mathcal{X}$ and a probability distribution $P(X)$ where $X = \{x_1, x_2, ..., x_n\} \in \mathcal{X}$. A Task, $\mathcal{T} = \{\mathcal{Y}, f(x)\}$, consists of a label space $\mathcal{Y}$ and a predictive function $f : \mathcal{X} \to \mathcal{Y}$, which can be learned from the data. Furthermore, we define a Source Domain, $D_S = \{(x_1^S, y_1^S), ..., (x_{n_S}^S, y_{n_S}^S)\}$ where $(x_i^S, y_i^S) \in (\mathcal{X}_S, \mathcal{Y}_S) \quad \forall i$, and a

Target Domain, $D_T = \{(x_1^T, y_1^T), ..., (x_{n_T}^T, y_{n_T}^T)\}$ where $(x_i^T, y_i^T) \in (\mathcal{X}_T, \mathcal{Y}_T) \quad \forall i$. Each Domain is accompanied by a corresponding Task: $\mathcal{T}_S = \{\mathcal{Y}_S, f_S(x)\}$ for the Source Task and $\mathcal{T}_T = \{\mathcal{Y}_T, f_T(x)\}$ for the Target Task.

Transfer learning can be defined as the situation where a Source Domain D_S with a Task $\mathcal{T}_S$ and a Target Domain D_T with a Task $\mathcal{T}_T$ are given, and the objective is to derive a Target Domain labeling function $f(\cdot)_T$ using the Source Domain D_S, even when $D_S \neq D_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. This describes a single Source setting, however, the notation can be trivially extended to multiple Sources. Since our focus is on homogeneous transfer learning, we place the restriction, $\mathcal{X}_S = \mathcal{X}_T$ and $\mathcal{Y}_S = \mathcal{Y}_T$. Note that this second restriction, $\mathcal{Y}_S = \mathcal{Y}_T$, may be slightly relaxed for classification problems where the Target Domain includes an unseen class. Therefore, there are 2 main ways that the Source and Target Domain can differ in transfer learning. First, the marginal probabilities are different, but the conditional probabilities are the same: $P(X_S) \neq P(X_T)$ and $P(Y_S|X_S) = P(Y_T|X_T)$. This is commonly referred to as *covariate shift*. Second, the marginal probabilities and the conditional probabilities are different: $P(X_S) \neq P(X_T)$ and $P(Y_S|X_S) \neq P(Y_T|X_T)$. This is commonly referred to as *sample selection bias*. The case where $P(Y_S) \neq P(Y_T)$ is often referred to as *class imbalance* for classification tasks and is such a common topic in traditional machine learning, that it is rarely talked about in a transfer learning setting. *Domain adaptation*, a phrase often used in the field, is a broad term used to define the transfer learning goal and has become synonymous with transfer learning as a whole. A brief summary of the notation can be found in Table 1.

A common example used to explain the notation is sentiment classification in Natural Language Processing. Let us assume that our goal is to create a binary sentiment classifier for movie reviews. Thus, our feature space is the movie review text (modeled as a bag-of-words for this example) and our label space is $\{0, 1\}$, where 1 implies a good review and 0 implies a bad review. Because human annotation is required to create labeled training data, we turn to other Domains for labeled data. If we took book reviews as our Source Domain, then we might expect the marginal probabilities to differ, but the conditional probabilities to be very similar. Thus we would have covariate shift with $P(X_S) \neq P(X_T)$ and $P(Y_S|X_S) = P(Y_T|X_T)$. This is because different words might show up in each review, such as "captivating" or "author" for book reviews and "stunning" or "director" for movie reviews, but even though the distribution of words differs for the different reviews, the words still have the same sentimental value. For example, "stunning" might be a rare word in a book review, but it still likely has a positive sentiment.

However, if we took refrigerator reviews as our Source Domain, then we might expect both the marginal and conditional distributions to differ. Then we would have sample selection bias with $P(X_S) \neq P(X_T)$ and $P(Y_S|X_S) \neq P(Y_T|X_T)$. This is because refrigerator reviews and movie reviews use differing vocabularies so the marginals are different and the vocabularies also have different sentimental meanings, thus the conditionals are different. An example of this would be “quiet”, which is likely a negative word for a movie, but a positive word for a refrigerator.

Table 1: Transfer Learning Notation

Notation	**Description**
$\mathcal{X}$	Feature Space
$\mathcal{Y}$	Label Space
D_S	Source Domain
D_T	Target Domain
$f(\cdot)$	Labeling Function
$P(X)$	Marginal Probability
$P(Y\|X)$	Conditional Probability
$P(Y)$	Label Probability

For the sake of continuity, we will now establish notation common throughout this survey in no particular order. The use of ' implies transpose and most, if not all cases of ^{T} imply the variable belongs to the Target Domain. Let $k(x, y) = \langle\phi(x), \phi(y)\rangle$ be a kernel function defined as the inner product of the feature transformations, ϕ, of x and y. Let K be the Kernel matrix such that $K_{ij} = k(x_i, x_j)$. Furthermore, let K_{D_1,D_2} be the Kernel matrix where $x_i \in D_1$ and $x_j \in D_2$ for Domains 1 and 2. Then let $K \in \mathbb{R}^{(n_T+n_S)\times(n_T+n_S)}$ be the Kernel matrix such that

$$K = \begin{bmatrix} K_{S,S} & K_{S,T} \\ K_{T,S} & K_{T,T} \end{bmatrix}.$$

Additionally, let L be a normalizing matrix such that

$$L_{ij} = \begin{cases} \frac{1}{n_S^2} & x_i, x_j \in X_S \\ \frac{1}{n_T^2} & x_i, x_j \in X_T \\ -\frac{1}{n_T n_S} & \text{otherwise} \end{cases} \qquad \forall i, j = 1, ..., n_S + n_T,$$

which will be useful for many calculations later on. Additionally, using common notation, H is the centering matrix, $H = I - \frac{1}{n}\mathbf{11}'$, and $\mathcal{L}$ is the Laplacian matrix, $\mathcal{L} = D - M$, where D is the degree matrix and M is the affinity matrix. Non-specified hyper-parameters will include, λ, γ, θ and μ. $J(\cdot)$ is a general risk function, $l(\cdot)$ is a general loss function, $f(\cdot)$ is a labeling function, $\|\cdot\|_F$ is the Frobenius norm, and $\|\cdot\|_K$ is the K-norm. While this is the general notation, some variables may be redefined for certain examples.

For the multi-Source setting with k Source Domains, let $D_s = \{(x_1^s, y_1^s), ..., (x_{n_s}^s, y_{n_s}^s)\}$ be s^{th} Source

Domain, where $s = 1, ..., k$. For scenarios where the Target Domain has labeled and unlabeled data, we will define the first n_T^l examples as labeled and the next n_T^u examples as unlabeled. Thus $D_T = \{(x_1^T, y_1^T), ..., (x_{n_T^l}^T, y_{n_T^l}^T), (x_{n_T^l+1}^T), ..., (x_{n_T}^T)\}$. When referencing distributions from a certain Domain, we will use $P(X_D)$ and $P_D(X)$ interchangeably, to imply the probability distribution of Domain D, however, for $P_D(X)$, we could pass an example from a different Domain to this probability to get the density of this example if it was distributed under D. It should be noted that equations taken from different sources may be formatted with the preceding notation, regardless of the original formatting.

3 Methods

3.1 Classification of Methods

There are two main ways to categorize transfer learning approaches: solution-based categorization and problem-based categorization. Problem-based categorization breaks methods into Inductive and Transductive. Inductive is the scenario where the Source Domain and the Target Domain have labels (but the Target Domain only has a few labeled instances). Transductive means that only the Source Domain has label information while only unlabeled data is available in the Target Domain.

Solution-based categorization breaks methods into instance-based, feature-based, parameter-based, and relational-based. Instance-based methods involve weighting the Source data and then adding it to the Target Domain's training set. Feature-based methods involve finding a feature representation where the Source Domain and the Target Domain are similar. Parameter-based methods look into transferring knowledge through model parameters. Relational-based methods involve finding some relationship between the Domains and usually require domain expertise and non-general solutions. We include a separate section on deep transfer learning due to recent success in this field, however, most, if not all, deep transfer learning is technically feature or parameter-based.

In this survey, we separate methods based on the solution categorization as it provides more information than the problem-based categorization. In fact, most methods assume the Transductive setting and can easily be reformed to fit the Inductive setting.

Due to the diverse range of problems and solutions that transfer learning addresses, there is no single common practice for comparing approaches. Some methods require labeled Target data, multiple Source Domains, or are tailored to specific areas such as NLP or Computer Vision. Consequently, comparing methods can be quite challenging. Additionally, literature that provides comparisons between multiple methods is limited. For this reason, we only include an empirical comparison for a few of the following methods.

3.2 Instance-Based

Instance-based transfer learning attempts to weigh the Source Domain data such that it follows the Target Domain's marginal distribution. For the majority of these methods, we assume that the conditional distributions are the same, but the marginal distributions are different: $P_S(X) \neq P_T(X)$ and $P_S(Y|X) = P_T(Y|X)$. We must also assume that the support of X_T is fully contained by the support of X_S. Once we have made these assumptions, we simply need to find a weighting function that converts the Source Domain's marginal distribution into that of the Target Domain. More formally, we need to find a function $w : \mathcal{X} \to \mathbb{R}$, such that $w(\cdot)$ satisfies the following equation.

$$P_T(X) = w(X) \cdot P_S(X)$$

$$w(X) = \frac{P_T(X)}{P_S(X)}$$

As seen above, the weighting function can be trivially solved for by rearranging the original equation. This means that the weighting function is simply the ratio of the marginal densities at a given input. However, since we don't have access to the actual densities of our sampled data, we resort to estimating the densities with the given samples. All instance-based transfer learning techniques either explicitly or implicitly attempt to calculate the ratio of the marginal distributions to weight the data.

Once an estimate has been formed, the Source data can be added to any available training data and treated as Target data as long as its training example is weighted. This means that for a given loss function $l(\cdot)$ instead of simply optimizing the mean of $l(\cdot)$ over all training data, you optimize the weighted average of $l(\cdot)$

$$L = \frac{1}{n} \sum_{i=1}^{n} w(x_i) \cdot l(x_i, y_i).$$

Where $w(x_i) = 1 \quad \forall x_i \in D_T$.

Shimodaira *et al.* [1] were one of the first to develop this framework to combat covariate shift. Their approach involves both normality assumptions and non-parametric kernel density estimation to calculate $P_T(X)$ and $P_S(X)$. They show that this method, Weighted Maximum Likelihood Estimation (WMLE), is optimal in the limit. The authors also perform many experiments that demonstrate the method's effectiveness, using simulated data. However, there are two main issues with their method. First, the simulations only involve small dimensional feature spaces, while kernel density estimation is known to perform poorly in high dimensions. Second, computing the densities separately induces a highly variable weight ratio, especially in a data-scarce environment. Thus estimating the ratio requires more sophisticated methods.

To fully understand many of the following methods in instance-based transfer learning, we must first look at a very influential paper in the field, that creates a new metric to compare distribution similarity. In 2006 Borgwardt *et al.* [2] introduced their method called Maximum Mean Discrepancy (MMD). MMD is the maximum distance between two distributions in a Reproducing Kernel Hilbert Space (RKHS). Let F be the set of all functions $f : \mathcal{X} \to \mathbb{R}$ on the unit ball in the universal Reproducing Kernel Hilbert Space. Then MMD is defined as

$$\text{MMD}(F, P_T, P_S) = \sup_{f \in F} \Big(E_{P_T}[f(x_T)] - E_{P_S}[f(x_S)] \Big),$$

which has the discrete, finite sample estimator

$$\text{MMD}(F, P_T, P_S) = \sup_{f \in F} \Bigg(\frac{1}{n_T} \sum_{i=1}^{n_T} f(x_T) - \frac{1}{n_S} \sum_{i=1}^{n_S} f(x_S) \Bigg).$$

If we let $f(x) = \langle \phi(x) \rangle_{\mathcal{H}}$ be the inner product of a feature transformation $\phi : \mathcal{X} \to \mathcal{H}$ onto a Hilbert Space and $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle_{\mathcal{H}}$, then we can utilize the kernel trick to estimate MMD. It is worth noting that the Gaussian kernel, $k(x_1, x_2) = \exp(\frac{-\|x_1 - x_2\|^2}{2\sigma^2})$, is one such kernel that is commonly used to compute this metric. Then we can write the finite sample estimation of MMD^2 as

$$\begin{aligned} \text{MMD}^2(F, P_T, P_S) = \frac{1}{n_T(n_T - 1)} \sum_{i \neq j}^{n_T} k(x_i^T, x_j^T) + \frac{1}{n_S(n_S - 1)} \sum_{i \neq j}^{n_S} k(x_i^S, x_j^S) \\ - \frac{2}{n_S n_T} \sum_{i=1}^{n_T} \sum_{j=1}^{n_S} k(x_i^T, x_j^S). \end{aligned}$$

Using the matrix notation we setup earlier, we can rewrite this summation into the compact form,

$$\text{MMD}^2(F, P_T, P_S) = \text{tr}(KL)$$

with K as the Kernel matrix for both Domains and L as the normalizing matrix that contains the terms outside of the summations. The authors further prove that this statistic, MMD^2, is asymptotically Gaussian. We now have a relatively fast and accurate way to non-parameterically compare distributions.

One of the first direct applications of this theory to transfer learning came from Huang *et al.* [3], including several authors from [2]. They use a technique called Kernel Mean Matching (KMM) which uses the equation

$$\begin{aligned} w(x) = \arg\min_{w} \Big\| E_{x \sim P_T(x)} \big[\Phi(x)\big] - E_{x \sim P_S(x)} \big[w(x)\Phi(x)\big] \Big\| \\ \text{s.t. } w(x) \geq 0 \text{ and } E_{x \sim P_S(x)}[w(x)] = 1. \end{aligned}$$

Solving this equation yields the estimate of $w(x) = \frac{P_T(x)}{P_S(x)}$. The finite sample estimation of this method can be written as

$$\begin{aligned} w(x) &= \arg\min_w \left\| \frac{1}{n_S} \sum_{i=1}^{n_S} w_i \cdot \Phi(x_i^S) - \frac{1}{n_T} \sum_{i=1}^{n_T} \Phi(x_i^T) \right\| \\ &= \arg\min_w \quad \frac{1}{n_S^2} W' K W - \frac{2}{n_S^2} \kappa' W + const, \end{aligned}$$

where $k(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle$, $K_{ij} = k(x_i^S, x_j^S)$ and $\kappa_i = \frac{n_S}{n_T} \sum_{j=1}^{n_T} k(x_i^S, x_j^T)$. We can then formulate the quadratic program

$$\begin{aligned} &= \min_w \frac{1}{2} W' K W - \kappa' W \\ \text{s.t. } \; w_i \in [0, w_{max}] \text{ and } &\left| \sum_{i=1}^{n_S} w_i - n_S \right| \leq n_S \epsilon. \end{aligned}$$

The hyper-parameters w_{max} and ϵ govern two aspects of the method, ensuring that $w(x) \cdot P_S$ forms a probability distribution and controlling the extent of the discrepancy between the means.

Dai *et al.* [4] go another route and attempt to estimate the weight ratio using Boosting. They modify the AdaBoost algorithm into an algorithm for transfer learning, calling it TrAdaBoost. Given a labeled Source data set and a labeled Target data set (which is assumed to be too small for traditional machine learning), TrAdaBoost iteratively updates the weights for all training examples, including the examples from the Target data set. TrAdaBoost follows the AdaBoost procedure for Target examples by increasing the weight of incorrectly classified examples. Conversely, TrAdaBoost decreases the weight of incorrectly classified Source examples because they are deemed to be far away from the Target distribution. A modified version of the original TrAdaBoost algorithm is shown in Algorithm 1. This method can easily be extended to a multi-Source setting, as seen in [5]. It is worth noting that when no Source data is passed to this algorithm, TrAdaBoost reduces to AdaBoost.

Other distribution comparison metrics can also be used to estimate the weight ratio. For example, Sugiyama *et al.* [6] estimate the ratio by minimizing the Kullback-Leibler (KL) Divergence between the Target marginal distribution and the weighted Source marginal distribution. KL Divergence between Domains can be written as

$$\begin{aligned} \mathrm{KL}\Big(P_T(X) \parallel w(X) \cdot P_S(X)\Big) &= \int_x P_T(x) \cdot \log\Big(\frac{P_T(x)}{w(x) \cdot P_S(x)}\Big) dx \\ &= \int_x P_T(x) \cdot \log\Big(\frac{P_T(x)}{P_S(x)}\Big) dx - \int_x P_T(x) \cdot \log w(x) dx. \end{aligned}$$

Since the first term is independent of $w(x)$ we can turn our attention to the second term. Furthermore,

Algorithm 1 TrAdaBoost

```
Input: $D_S = \{(x_1^S, y_1^S), ..., (x_{n_S}^S, y_{n_S}^S)\}$ and $D_T = \{(x_1^T, y_1^T), ..., (x_{n_T}^T, y_{n_T}^T)\}$ where $y \in 0, 1$
        and Weak Learner: $h(\cdot)$ where $h : \mathcal{X} \to [0, 1]$
Initialize: $w_S^1 = (w_1^1, ...w_{n_S}^1)$ and $w_T^1 = (w_1^1, ...w_{n_T}^1)$ where $w^1 = [w_S, w_T]$
for $t = 1, ..., N$ do
    1. Normalize $w^t$
    2. Train $h_t(\cdot)$ on $D_S \cup D_T$ using $w^t$
    3. Calculate the error of $h_t$ on $D_T$
           $\epsilon_t = \sum_{i=1}^{n_T} \frac{[w_T]_i^t \cdot |h_t(x_i^T) - y_i^T|}{\sum_1^{n_T} [w_T]_i^t}$
    4. Set $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ and $\beta = \frac{1}{\sqrt{2 \cdot \ln \frac{n_S}{N}}}$
    5. Update the weight vector
           $[w_S]_i^{t+1} = w_i^t \beta^{|h_t(x_i^S) - y_i^S|}$
           $[w_T]_i^{t+1} = \beta_t^{-|h_t(x_i^T) - y_i^T|}$
end for
Output: Hypothesis
           $H(x) = \begin{cases} 1, & \prod_{t=\frac{N}{2}}^{N} \beta_t^{-h_t(x)} \geq \prod_{t=\frac{N}{2}}^{N} \beta_t^{-\frac{1}{2}} \\ 0, & \text{otherwise} \end{cases}$
```

let us specify the functional form of an estimate for $w(x)$, $\hat{w}(x)$,

$$\hat{w}(x) = \sum_{\ell}^{b} \alpha_\ell \cdot \varphi_\ell(x),$$

where $\{\alpha_\ell\}_{\ell=1}^b$ are learnable parameters and $\{\varphi_\ell\}_{\ell=1}^b$ are basis functions such that $\varphi_\ell(x) \geq 0 \quad \forall\ \ell \in \{1, 2, ..., b\}$. For simplicity, the authors choose the Gaussian Kernel, $k(x_1, x_2) := \exp\left(-\frac{\|x_1 - x_2\|}{2\sigma^2}\right)$, with respect to b points in the Target Domain, each denoted by c_ℓ, $\ell = 1, ..., b$. Thus we have

$$\hat{w}(x) = \sum_{\ell}^{b} \alpha_\ell \cdot k(x, c_\ell).$$

Therefore our minimization can be discretized and turned into

$$\max_{\{\alpha_\ell\}} \left[\sum_{j=1}^{n_T} \log\left(\sum_{\ell=1}^{b} \alpha_\ell \cdot k(x_j^T, c_\ell \right) \right]$$

$$\text{s.t.} \quad \sum_{i=1}^{n_S} \sum_{\ell=1}^{b} \alpha_\ell \cdot k(x_i^S, c_\ell) \ = n_S \text{ and } \alpha_1, \alpha_2, ..., \alpha_b \geq 0.$$

The optimal $\hat{w}(x)$ can be found by solving the convex optimization problem using gradient ascent. This method can be shown to be asymptotically normal around the optimal weight ratio. Additionally, it is worth noting that while others, including [3] estimate discrete weights for each Source example, this method estimates a functional form for $w(x)$. This is extremely beneficial for hyper-parameter selection methods such as Cross-validation.

Using clever manipulation of the weight ratio, it can be shown that $\frac{P_T(x)}{P_S(x)} = \frac{P(x \in D_S)}{P(x \in D_T)} \cdot \big(\frac{1}{P(x \in D_S|x)} - 1\big)$. Bickel *et al.* [6] derive this equation and develop it into a new estimation technique for the weights. In their method, we can estimate the weight ratio by training a discriminative model to predict the probability of the sample residing in the Target Domain. However, instead of fitting this Domain classifier and then fitting another classifier for $P(Y|X, W)$ the authors solve both problems at once using a joint MAP approach,

$$[w, v]' = \arg\max_{w,v} P(w, v, |D_S, D_T),$$

where w is the weight ratio vector and v is the parameter vector for the label classifier. The MAP equation can be factored into

$$[w, v]' = \arg\max_{w,v} \left(\prod_{i=1}^{n_S} P(y|x_i^S, w)^{\frac{1}{P(x \in D_S|x_i^S, v)} - 1} \right)^{\frac{P(x \in D_S|v)}{P(x \in D_T|v)}}$$
$$\cdot \left(\prod_{i=1}^{n_S} P(x \in D_S|x_i^S, v) \cdot \prod_{i=1}^{n_T} P(x \in D_T|x_i^T, v) \right) \cdot P(w) \cdot P(v).$$

The authors use logistic functions for both sets of likelihoods and exponential priors on the parameters w and v. The estimated parameters can be found via Newton's method optimization.

We can also relax some of the initial constraints we made for instance-based methods. Jiang *et al.* [7] attempt to relax constraints that the conditional densities of the two Domains have to be equal, $P_S(Y|X) \neq P_T(Y|X)$. They do this by adding two more instance weights, α_i and $\gamma_i(y)$ in addition to w. Here α_i represents how close the conditional densities, $P_T(y_i^S|x_i^S)$ and $P_S(y_i^S|x_i^S)$, are at x_i^S, with larger values of α_i indicating the densities are close. Additionally, $\gamma_i(y)$ indicates how likely we are to assign the label y to a given unlabeled Target Domain data point, x_i^T, and include the pseudo-labeled set, (x_i^T, y) in our training data. Then the MAP estimate becomes

$$\hat{\theta} = \arg\max_{\theta} \quad \lambda_1 \cdot \frac{1}{\sum_{i=1}^{n_S} \alpha_i w_i} \sum_{i=1}^{n_S} \alpha_i w_i \log P(y_i^S|x_i^S, \theta) + \lambda_2 \cdot \frac{1}{n_T^l} \sum_{i=1}^{n_T^l} \log P(y_i^T|x_i^T, \theta)$$
$$+ \lambda_3 \cdot \frac{1}{\sum_{k=1+n_T^l}^{n_T} \sum_{y \in \mathcal{Y}} \gamma_k(y)} \sum_{i=1+n_T^l}^{n_T} \sum_{y \in \mathcal{Y}} \gamma_k \log P(y|x_k^T, \theta) + \log p(\theta).$$

The authors choose α_i by setting k of the α_i, corresponding to the highest confidence incorrectly classified Source data, to 0 and the rest of the α_i to 1. They set all w_i to 1 and k of the γ_i are set to

1 while the rest are set to 0, using a bootstrapping method.

Sun *et al.* [8] also assume the conditional densities differ and create a 2 stage framework for multiple Source Domains. The first stage involves using the KMM technique found in [3] to get the weight ratios for each example in each Source Domain. For the second stage, let $H_i^S = [h_i^1, ...h_i^k]$ be the predicted labels from the k Source Domains for the i^{th} instance of the unlabeled Target data. Let $\beta = [\beta_1, ..., \beta_k]$ be the weight given to the k^{th} Source Domain. Finally, let W_{ij} be the similarity between the i^{th} and the j^{th} unlabeled Target instance. Then, the second stage minimization problem is

$$\min_{\beta} \sum_{i,j=1}^{n_T} (H_i^S \beta - H_j^S \beta)^2 W_{ij}.$$

This corresponds to a smoothness constraint, where similar values of x_T must be given similar values of $P(y|x_T)$ with the estimated conditional distribution.

When the constraint of equal conditional densities does not hold, instance weighting becomes significantly more complex. The authors of [7] had to make several simplifications to estimate the weights, highlighting the added complexity when the constraint of equal conditional densities is not satisfied. In fact, they even assigned one set of weight ratios to a fixed value of 1 to aid in the estimation of their remaining weights. Therefore, to tackle the problem of conditional density differences, we turn to feature-based transfer learning.

3.3 Feature-Based

Feature-based transfer learning involves mapping the original feature space into a new feature space where the Domains are more aligned. This can take the form of feature augmentation, feature selection, and feature transformation. Instance-based methods have 3 main issues that feature-based methods attempt to fix. The first is that the conditional densities usually have to be the same for instance-based methods, while they don't for feature-based methods. The second is that instance weighting is a challenging task in high-dimensions, while feature mapping is well suited for high-dimensional data. Finally, instance weighting requires the support of the Target Marginal to be contained in the Source Marginal, but feature mapping makes no such requirements. Feature-based methods can be further categorized into Symmetric and Asymmetric methods.

Asymmetric methods aim to transform the feature space of the Source Domain into that of the Target Domain. This does not necessarily mean that there are no feature transformations in the Target Domain, it simply means that any transformations in the Target Domain are not attempting to bring the Domain towards that of the Source Domain. For example, running PCA on the Target Domain and then projecting the Source Domain to these principal components would be considered asymmetric.

These methods require the assumption that after the transformations are applied to the Source Domain, the Marginals, and Conditionals of the Domains are the same. That means, if $\phi(\cdot)$ is the feature transformation, then $P(X_T) = P(\phi(X_S))$ and $P(Y|X_T) = P(Y|\phi(X_S))$.

Symmetric methods attempt to transform the feature space of both the Source and the Target into a new feature space where the Domains are similar. For these methods, we must assume that in this new feature space, the marginal and conditional distributions are the same. That is, $P(\phi(X_T)) = P(\phi(X_S))$ and $P(Y|\phi(X_T)) = P(Y|\phi(X_S))$.

Pan *et al.* [9] develop a symmetric feature-based method that seeks to minimize the MMD of the Domains, while also maximizing the variance of the data. Let $K \in \mathbb{R}^{(n_T+n_S)\times(n_T+n_S)}$ be the Kernel matrix of both Domains, induced by a universal kernel k, and let L be the normalizing matrix seen earlier. Then the optimization problem is

$$\begin{aligned} K^* = \arg\min_{K} &\quad \mathrm{tr}(KL) - \lambda \mathrm{tr}(K) \\ \text{s.t.} \quad K_{ii} + K_{jj} - 2K_{i,j} &= d_{ij}^2 \quad \forall (i,j) \in \mathcal{N} \\ K \cdot \mathbf{1}_{n_T+n_S} &= -\epsilon \cdot \mathbf{1}_{n_T+n_S}, \end{aligned}$$

where ϵ is a small positive constant and d_{ij}^2 is the distance between the Nearest Neighbors, x_i and x_j. Here, the first trace corresponds to minimizing MMD and the second trace supports a high variance in the embedding. After obtaining K^*, PCA can be performed to decrease the dimensions of the embedding space. A model can be trained on the Source data in the new feature space and, if our assumptions hold, the model should perform well in the Target Domain. While intuitive and simple, this method has huge computational costs, with a time complexity of $O(n^{6.5})$. Thus, others have looked into similar methods that are much more efficient.

Duan *et al.* [10] use a related technique for finding a symmetric embedding space. They also attempt to minimize MMD, but instead of maximizing the variance of the embedding, they simultaneously minimize the structural risk of a classifier. Their resulting objective function is

$$\min_{k,f} \frac{1}{2}\mathrm{tr}(KL)^2 + \theta R(k,f).$$

They further assume that the kernel, k that induces K, is a linear combination of other fixed kernels, such that $k = \sum_{m=1}^{M} d_m k_m$. Let $p = [p_1, ..., p_M]'$ where $p_m = \mathrm{tr}(K_m L)$ and k_m induces K_m. Then the objective function simplifies to

$$\min_{d,f} \frac{1}{2} d' p p' d + \theta R(d,f).$$

Thus, they only need to estimate the weights of the kernels and the parameters of the classifier, vastly simplifying the minimization problem in [9].

Akin to the examples above, Baktashmotlagh *et al.* [11] create a method called Domain Invariant Projection (DIP), and its extension, Domain Invariant Projection with Co-Clustering (DIP-CC). The method works by minimizing the MMD of the Source and Target Domains after they are linearly projected into a d dimensional subspace. This corresponds to minimizing

$$D(W'X_T, W'X_S) = \left\| \frac{1}{n_S} \sum_{i=1}^{n_S} \phi(W'x_i^S) - \frac{1}{n_T} \sum_{i=1}^{n_T} \phi(W'x_i^T) \right\|.$$

This method can be formatted using the kernel function $k(x_i, x_j) = k_W(W'x_i, W'x_j)$ where the Kernel matrix induced by k_W is K_W. Thus the minimization problem can be written as

$$\begin{aligned} W^* = \arg\min_W \quad & \operatorname{tr}(K_W L) \\ \text{s.t.} \quad & W'W = I_d. \end{aligned}$$

This is the DIP formulation and can be solved using conjugate gradient descent. This can also be extended to encourage similar Source labels to cluster together in the new feature space.

$$\begin{aligned} W^* = \arg\min_W \quad & \operatorname{tr}(K_W L) + \lambda \sum_{c=1}^{C} \sum_{i=1}^{n_c} \left\| W'(x_{i,c}^S - \mu_c) \right\| \\ \text{s.t.} \quad & W'W = I_d \end{aligned}$$

This second term enforces co-clustering in the Source Domain such that examples of the same class, c are relatively close to one another.

Pan *et al.* [12] develop a similar technique that improves on the computational costs of [9]. They introduce Transfer Component Analysis (TCA) for an unsupervised and a supervised Target Domain. For the unsupervised version (UTCA), they start with the objective function in [9], but introduce a transformation matrix, $\tilde{W} \in \mathbb{R}^{(n_T+n_S)\times m}$ that transforms the features into an m dimensional space. The new kernel can then be computed as $\tilde{K} = KWW^TK$, where $W = K^{-\frac{1}{2}\tilde{W}}$. In contrast to [9], this technique avoids the need for PCA to reduce the dimensionality of the embedding post-minimization. The authors also seek to reduce the complexity of the embedding by keeping $\operatorname{tr}(W^TW)$ small. Thus their objective function is

$$\begin{aligned} \min_W \operatorname{tr}(W'KLKW) + \mu \operatorname{tr}(W'W) \\ \text{s.t.} \quad W'KHKW = I_m, \end{aligned}$$

where H is the centering matrix. To extend this to the supervised setting (SSTCA), we switch $\tilde{K}$

with $\tilde{K}_{yy} = \gamma K_l + (1-\gamma)I$, where $[K_l]_{ij} = k_{yy}(y_i, y_j)$ if $y_i, y_j \in D_S$ otherwise $[K_l]_{ij} = 0.$. Here γ is a trade-off parameter between label dependence and maximal variance. Additionally, in order to perverse the locality of the embedding, let $\mathcal{L}$ be the graph Laplacian matrix. Then the minimization problem is

$$\begin{aligned} \min_W \operatorname{tr}(W'KLKW) + \mu\operatorname{tr}(W'W) + \frac{\lambda}{(n_S+n_T)^2}\operatorname{tr}(W'K\mathcal{L}KW) \\ \text{s.t.} \quad W'KH\tilde{K}_{yy}HKW = I_m, \end{aligned}$$

where the new term in the objective function preserves locality and the new $\tilde{K}_{yy}$ encourages class label information to hold.

Long *et al.* [13] introduce a similar method to [12], called Joint Distribution Adaptation (JDA). They take the maximization objective of PCA, $\operatorname{tr}(A'XHXA)$, and extend it to minimize MMD, $\operatorname{tr}(A'XLXA)$. They then attempt to use pseudo-labels for the Target Domain to minimize the difference between $P_S(\phi(x_S)|y=c)$ and $P_T(\phi(x_T)|y=c)$. Let $D_S^{(c)}$ be the set of all examples in D_S that belong to class $c \in \{1, ..., C\}$ and $D_T^{(c)}$ be the set of all examples in D_T that are given the pseudo-label c. Finally, let L_c be the matrix such that

$$[L_c]_{ij} = \begin{cases} \frac{1}{n_S^{(c)} n_S^{(c)}}, & x_i, x_j \in D_S^{(c)} \\ \frac{1}{n_T^{(c)} n_T^{(c)}}, & x_i, x_j \in D_T^{(c)} \\ -\frac{1}{n_S^{(c)} n_T^{(c)}}, & \begin{cases} x_i \in D_S^{(c)}, x_j \in D_T^{(c)} \\ x_j \in D_S^{(c)}, x_i \in D_T^{(c)} \end{cases} \\ 0, & \text{otherwise} \end{cases}.$$

Then the objective function becomes

$$\min_{A^T XHX^T A = I} \sum_{c=0}^{C} \operatorname{tr}(A'XL_cX'A) + \lambda \parallel A \parallel_F^2,$$

where L_0 corresponds to the original normalization matrix, L. This can be trivially expanded to include Kernel PCA as the embedding technique. It is worth noting that UTCA [12] is a special case of JDA with C=0.

Long *et al.* [14] extend this method to iteratively pseudo-label the data by simultaneously training a classifier, $f(x)$ using the following equation

$$\alpha = \arg\min_{\alpha} \left[\sum_{i=1}^{n_S} \ell(f(x_i^S), y_i^S) \right] + \sigma \mathrm{tr}(\alpha' K \alpha) + \lambda \mathrm{tr}(\alpha' K L K \alpha)$$
$$+ \left[\mu \sum_{c=1}^{C} \mathrm{tr}(\alpha' K L_c K \alpha) \right] + \gamma \mathrm{tr}(\alpha' K \mathcal{L} K \alpha),$$

where α parameterizes the classifier, the first two terms form the structural risk of the classifier, the third term is the difference in the Marginals, the fourth term is the difference in the Conditionals ($P(x|y = c)$), and the last term enforces locality. $\mathcal{L}$, as before, is the graph Laplacian matrix.

While MMD is the most common distance metric used for feature-based methods, there are examples of other measures being used. Si *et al.* [15] replace the minimization of MMD with Bregman Divergence, calling the new method Transfer Subspace Learning (TSL). Empirical Bregman Divergence in this instance is defined as

$$D_W(P_S \parallel P_T) = \frac{1}{n_S^2} \sum_{i,j=1}^{n_S} G_{\Sigma_{11}}(z_i - z_j) + \frac{1}{n_T^2} \sum_{i,j=1}^{n_T} G_{\Sigma_{22}}(z_i - z_j) - \frac{2}{n_S \cdot n_T} \sum_{i=1}^{n_S} \sum_{j=1}^{n_T} G_{\Sigma_{12}}(z_i - z_j),$$

where $z_i = W' x_i$, G_Σ defines a Gaussian kernel with variance Σ, and Σ_1, Σ_2 are arbitrary variances where $\Sigma_{11} = \Sigma_1 + \Sigma_1$, $\Sigma_{12} = \Sigma_1 + \Sigma_2$ and $\Sigma_{22} = \Sigma_2 + \Sigma_2$. We can see that this form bears resemblance to the MMD^2 empirical estimator. Thus for some chosen embedding objective function $F(W)$ that returns a linear projection matrix W, the authors' method can be generally defined as

$$W = \arg\min_{W} F(W) + \lambda D_W(P_S \parallel P_T).$$

The researcher's experiments seem to show that using Bregman Divergence works better than MMD in some cases.

Some feature-based techniques don't use any explicit measure of distribution similarity. Duamé *et al.* [16] create a remarkably simple technique for symmetric feature-based learning. They duplicate the original feature space into 3 distinct sections, a general section, a Source specific section, and a Target specific section. The Target and Source feature spaces have 0s in the other's section such that if the feature mapping is $\Phi(\cdot) = \langle General, Source, Target \rangle$, then the Source mapping is $\Phi_S(x) = \langle x, x, 0 \rangle$ and the Target mapping is $\Phi_T(x) = \langle x, 0, x \rangle$. A model can be trained on this feature space in both Domains and will naturally pick out the features that should be in the General section by giving them more weight.

Quattoni *et al.* [17] propose a novel method to create a new feature space for image classification. They first perform Singular Value Decomposition (SVD) on $K_{T,T}$ and create a projection matrix A

using all of the non-zero eigenvalues. The new representation is then, $z = A'\varphi(x)$, where $\varphi(x) = [k(x, x_1^T), ..., k(x, x_{n_T}^T)]'$. The columns of A are referred to as prototypes.

Then for each Source, s, a classifier can be trained in the prototype space such that

$$f_s(x^s) = w_s' z(x^s)$$
$$\min_w \quad \lambda \sum_{i=1}^{n_s} l(f_s(x_i^s), y_i^s) + \sum_{i=1}^{n_T} |w_i|.$$

The ℓ_1 norm induces sparsity in the model, essentially choosing the prototypes most suited for the task. Then, the optimal spare prototype space over all Sources is

$$\min_W \quad \sum_{s=1}^{k} \lambda_s \sum_{i=1}^{n_s} l(f_s(x_i^s), y_i^s) + \sum_{j=1}^{n_T} \max_s |W_{js}|,$$

where W is a weight matrix with W_{ij} corresponding to the i^{th} coefficient of the k^{th} Source. Then the second term is a combination of a ℓ_1 and a ℓ_∞ norm over the weights of the various models. This means that models are more heavily discouraged from adding a prototype if no other models are using it.

We can then choose the prototypes, $R = \{\ r : \max_s |W_{rs}^*|\ > \theta\}$, from W where more than θ Sources used the given prototype. A new representation can be formed by taking the columns of A that correspond to the column indices in R. Thus, we can project the Source data into this space and train a classifier that can perform well in the Target Domain.

Blitzer *et al.* [18] develop a method for Part of Speech (POS) tagging. They assume they have POS labels for the Source Domain and no labels for the Target Domain. First, they create m pivot feature, which are features for a given word such as <the word to the right> or whether a certain word is near that word. They then take all data and fit m linear predictors of the form, $f_i(x) = \text{sgn}(w_i \cdot x)$, which answers the question, "Does pivot feature i occur in this instance?". The feature itself is removed from x else $f(x)$ would be fully predictive based just on the given word. The weights are fit to minimize

$$\hat{w}_i = \arg\min_w \quad \sum_j l(w \cdot x_j, p_i(x_j)) + \lambda\|w\|.$$

By creating $W = [\hat{w}_1|...|\hat{w}_m]$, we have a matrix such that $\text{sgn}(W_{ij})$ is the direction of the correlation between the i^{th} feature and the j^{th} pivot feature. We can then perform SVD on W and choose the top components to get the projection matrix A. A POS classifier can then be trained on a concatenation of x_S and $A \cdot x_S$ that will perform well on the Target Domain.

Gong *et al.* [19] use a much different technique than the previous methods, called Geodesic Flow

Kernel (GFK). They first perform PCA on both Domains (PCA_S and PCA_T) and PCA on the combined data set of the Source and Target Domains (PCA_{S+T}). They then define the Subspace Disagreement Measure (SDM) as

$$\mathcal{D}(d) = 0.5[\sin \alpha_d + \sin \beta_d],$$

where α_d denotes the d^{th} principle angle between PCA_S and PCA_{S+T} and β_d between PCA_T and PCA_{S+T}. A small value implies PCA_S and PCA_T are aligned at the d^{th} dimension while a large value (of 1) implies the opposite. The authors compute the optimal number of dimensions as $d^* = \min\{d|\mathcal{D}(d) = 1\}$. They then construct the Geodesic Curve between PCA_S and PCA_T on the Grassmannian Manifold, $G(D, d^*)$. This infinite continuous interpolation between the Domains on the manifold can be parameterized using the kernel trick, such that the inner product of two points x_i and x_j at all points on the interpolation is $x_i' G x_j$, where G is the Geodesic Kernel. A classifier can then be trained using this kernel.

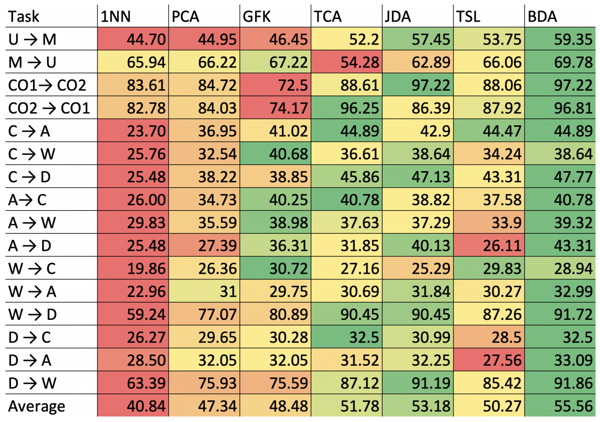
Shi *et al.* [20] take inspiration from metric learning and create a linear symmetric transformation for both Domains, parameterized by the projection matrix A. Their objective function has two parts. The first part, I_t, seeks to maximize the mutual information between the data and a pseudo-label, $\hat{Y}$, using a posterior for the pseudo-label given by the probability of each label, $\hat{p}$, using a Soft Nearest Neighbors classification technique in the embedding space. The second part, I_{st}, seeks to minimize the mutual information between the data and its Domain label, Q, using the posterior, $\hat{q}$ which is the probability of being assigned each Domain label using a Soft Nearest Neighbor classifier on all of the data in the embedding space. Their final objective function is

$$\begin{aligned} L^* = \arg\min_L & -I_t(X, \hat{Y}, A) + \lambda I_{st}(X, Q, A) \\ & \text{s.t.} \quad \mathrm{tr}(L^T L) \leq d, \end{aligned}$$

where the constraint controls the scale of the distances that are computed using A. Thus while the first term enforces the clusters of the Target Domain to be around clusters in the Source Domain, the second term enforces that these clusters be indistinguishable from one another.

Wang *et al.* [21] extend the method from [14] to a method called Balanced Distribution Adaptation (BDA). They introduce a weight to separate the importance of the Marginal similarity vs the Conditional similarity. They also perform the optimization iteratively, updating the pseudo-labels at each step. Finally, they add a term to account for class imbalance in the Source labels and Target pseudo-labels.

While most of the transfer learning literature does not use a single unified data set to compare methods, [21] has a table that includes 4 of the above transfer learning methods along with 2 controls, evaluated on image classification tasks. The classification algorithm used is 1NN, with the transfer learning methods (and PCA) used as dimensionality reduction techniques. The table notation is, <Source Domain> $\rightarrow$ <Target Domain>. The data sets are USPS (U) [22], MNIST (M) [23], COIL20 (CO) [24], Caltech-256 (C) [25], Office [26] (composed of 3 types of objects: Amazon (A), Webcam (W), and DSLR (D)). Figure 1 contains a reformatted version of this table. Note that the data is taken from [21] and thus the tests may be slightly biased towards a high performance for BDA.



Task	1NN	PCA	GFK	TCA	JDA	TSL	BDA
U → M	44.70	44.95	46.45	52.2	57.45	53.75	59.35
M → U	65.94	66.22	67.22	54.28	62.89	66.06	69.78
CO1→ CO2	83.61	84.72	72.5	88.61	97.22	88.06	97.22
CO2 → CO1	82.78	84.03	74.17	96.25	86.39	87.92	96.81
C → A	23.70	36.95	41.02	44.89	42.9	44.47	44.89
C → W	25.76	32.54	40.68	36.61	38.64	34.24	38.64
C → D	25.48	38.22	38.85	45.86	47.13	43.31	47.77
A→ C	26.00	34.73	40.25	40.78	38.82	37.58	40.78
A → W	29.83	35.59	38.98	37.63	37.29	33.9	39.32
A → D	25.48	27.39	36.31	31.85	40.13	26.11	43.31
W → C	19.86	26.36	30.72	27.16	25.29	29.83	28.94
W → A	22.96	31	29.75	30.69	31.84	30.27	32.99
W → D	59.24	77.07	80.89	90.45	90.45	87.26	91.72
D → C	26.27	29.65	30.28	32.5	30.99	28.5	32.5
D → A	28.50	32.05	32.05	31.52	32.25	27.56	33.09
D → W	63.39	75.93	75.59	87.12	91.19	85.42	91.86
Average	40.84	47.34	48.48	51.78	53.18	50.27	55.56

Figure 1: A comparison of feature-based methods using Accuracy (%) on 16 different tasks, evaluated using GFK [19], TCA [12], JDA [13], TLS [15], and BDA [21]. It is implied that the TLS method uses PCA as its dimensionality reduction algorithm, but never explicitly stated.

When choosing between featured-based methods it is important to consider each unique situation since no method is perfect. If your data set is large, maybe the computational considerations would discourage you from [9]. If you want a simple algorithm to pick out cross-domain features, [18] would be the best choice. If you hope to utilize cluster information in your embedding, then use one of [11], [12], or [13]. If you have multiple Sources, maybe choose [17]. All of these techniques have their own merit and the best method for a given problem depends heavily on the data and what assumptions the researcher is willing to make.

3.4 Parameter-Based

In parameter-based transfer learning, a model is fit on the Source Domain and knowledge is transferred to the Target Domain through model parameters. For example, a model can be fit on the Source Domain and then those parameter values can be used as Prior Probability modes for a model on the Target Domain.

Stark *et al.* [27] take a probabilistic approach and propose an object classifier for knowledge transfer. For the Source Domain, they split the joint posterior of the features and hypothesis, $H = [h_1, ..., h_p]$, into 4 parts (and a prior) based on a conditional independence assumption. Those parts are Local Shape features (S), Symmetric relation features (B), Layout features (X), and Relative Scale features (R).

$$\underbrace{P(X,R,S,B,H|\theta)}_{posterior} = \underbrace{P(S|H,\theta)}_{Local\,Shape} \cdot \underbrace{P(B|H,\theta)}_{Symm.\,Rel.} \cdot \underbrace{P(X|H,\theta)}_{Layout} \cdot \underbrace{P(R|H,\theta)}_{Rel.\,Scale} \cdot \underbrace{P(H|\theta)}_{Prior}$$

While this is a general framework, in their experiments, the authors use multiplied Gaussians to create each distribution. The optimal hypothesis, H is then $H_{MAP} = \arg\max_H P(H|\theta, X, R, S, B,)$ or equivalently $H_{MAP} = \arg\max_H P(X, R, S, B, H|\theta)$. A Metropolis-Hastings algorithm is used to draw samples from this distribution to find an estimate for H_{MAP}.

To then transfer knowledge to the Target Domain, we can take the parameters from the individual distributions based on the similarity of the Source and Target Domain. For example, if our Source classifier was trained on detecting horses and our Target classifier is for giraffes, we might only take the Local Shape and Symmetric Relation distributions from the Source to transfer to the Target (because they are both animals with a head and neck, but the layout and scale of the bodies are much different).

Bruzzone *et al.* [28] introduce an iterative method called Domain Adaptation Support Vector Machine (DASVM) that transfers knowledge through SVM parameters. The authors assume there are no labeled Target instances. Their algorithm is as follows.

1. Fit an SVM to training data (initially only Source data)
2. Classify all Unlabeled Target data and add the Target data that are (1) inside the Margin bands and (2) closest to the Margin bands to the training data
3. Retrain the SVM on the training data. If the predicted label of any Target data switches from the previous iteration, remove it from the training set and add it back to the unlabeled set.
4. Remove the Source data that are furthest away from the Separating Hyperplane
5. Repeat until Convergence

A depiction of this algorithm at different stages of training can be seen in Figure 2.

Tommasi *et al.* [29] and later Tommasi *et al.* [30] create a method for multi-Source transfer. They assume that there are k Sources, each with a Kernel SVM (K-SVM) classifier to predict -1 or 1 for the class associated with the given Source. The estimated parameters for the j^{th} Source are given by w_j. They also assume access to a small amount of labeled Target data. Thus, the objective function for the Target classifier can be written as

$$[W^*, b^*]' = \arg\min_{w,b} \frac{1}{2}\left\| w - \sum_{j=1}^{k} \beta_j w_j \right\|^2 + \frac{C}{2}\sum_{i=1}^{n_T^l} \zeta_i (y_i - w\phi(x) - b)^2,$$

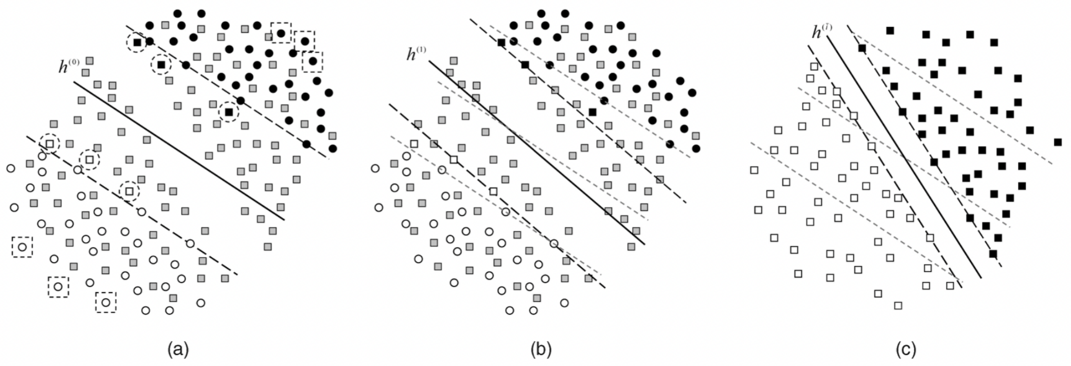


Figure 2: DASVM during training. Circles are Source examples, squares are Target examples. (a) depicts the first iteration of the algorithm while (b) and (c) show the algorithm moving towards converging. White and black examples are the training data while grey examples are the currently unlabeled data. The circled Target examples in (a) are being added to the training set and the boxed Source examples are being removed from the training set. We can see that at (c) there are no more Source examples in the training set. Taken from [28]

where W and b are the parameters of the K-SVM and $\|\beta\|_2 \leq 1$ controls the transfer from each Source model. β can be estimated by rearranging the objective function to minimize an upper bound on the Leave-One-Out Cross-validation error.

Yao *et al.* [5] extend the TrAdaBoost framework from [4] to a parameter-based method for multiple Sources. They first perform the traditional AdaBoost algorithm on each Source Domain and aggregate the collection of weak learners. Then they perform AdaBoost on the Target Domain, where at each step, a weak classifier from the set of Source weak classifiers is chosen that minimizes the weighted error on the Target Domain. This allows the algorithm to pick and choose what information it wants to transfer from the Source, based on the data's weights at the current iteration.

Chattopadhyay *et al.* [31] derive a general framework for parameter-based transfer. They assume the multi-Source setting with a limited number of labeled Target examples. The Target classifier is $f^T(x)$, where $f^T = [f(x_1^T), ..., f(x_{n_T}^T)]$, and is compared to a weighted combination of Source classifiers, $f^s(x)$. Their objective function is then

$$\min_{f^T \in \mathcal{H}_K} \gamma_A \|f^T\|_K^2 + \frac{1}{n_T^l} \sum_{i=1}^{n_T^l} (f_i^T - y_i^T)^2 + \frac{\theta}{2} \sum_{j=n_T^l+1}^{n_T} \|f_j^T - \sum_{s=1}^{k} \gamma_s f_j^S\|^2 + \frac{\gamma_I}{n_T^2} f^{T'} \mathcal{L} f^T.$$

The first term controls the complexity of the Target classifier, the second term computes the loss of the classifier on labeled Target data, the third term enforces the Target Classifier to be close to a weighted combination of Source classifiers, and the final term enforces nearby data points to have similar predictions (a manifold smoothness assumption). The weight for each Source classifier, γ_s, is found by optimizing over another smoothness assumption on the conditional probability.

Zhuang *et al.* [32] use Non-negative Matrix Tri-Factorization to perform transfer learning for document classification. They factor the feature matrix, $X_{m \times n}$, with n documents and m words, into

$$X_{m \times n} = F_{n \times k_1} \cdot S_{k_1 \times k_2} \cdot G'_{n \times k_2},$$

where X_{ij} represents the joint probability for the i^{th} word and the j^{th} document, F_{ij} represents the probability that the i^{th} word belongs to the j^{th} word cluster, S represents the association between word clusters and document clusters, and G_{ij} represents the probability that the i^{th} document belongs to the j^{th} document cluster. X can be trivially estimated by normalizing a data matrix $\tilde{X} \in \{0,1\}^{m \times n}$ such that $X = \frac{\tilde{X}}{\sum_{i,j} \tilde{X}_{ij}}$. The matrix factorization can be found for the Source Domain by solving

$$\begin{aligned} [F_S^*, S^*, G_S^*]' = \arg \min_{F_S, S, G_S} \quad & \left\| X_S - F_S \cdot S \cdot G'_S \right\|^2 + \frac{\lambda}{n_S} \left\| G_S - G_0 \right\|^2 \\ \text{s.t.} \quad & \sum_{j=1}^{k_1} [F_S]_{ij} = 1, \quad \sum_{j=1}^{k_2} [G_S]_{ij} = 1. \end{aligned}$$

where G_0 represents the actual labels for the Source Domain. By assuming the association between word clusters and document cluster is constant across Domains (S is the same across Domains), we can take S^* from the optimum of the above formulation and use it to solve the problem in the Target Domain.

$$\begin{aligned} [F_T^*, G_T^*]' = \arg \min_{F,G} \quad & \left\| X_T - F_T \cdot S \cdot G'_T \right\|^2 \\ \text{s.t.} \quad \sum_{j=1}^{k_1} [F_T]_{ij} = 1, & \quad \sum_{j=1}^{k_2} [G_T]_{ij} = 1. \end{aligned}$$

Here we only need to solve for F_T and G_T and then we can use G_T^* as our labels for the Target Domain. In practice, this sequential optimization is combined to form a single joint optimization problem and solved using an iterative optimization technique.

Wang *et al.* [33] use a very similar technique as in [32], but for image classification with some labeled Target data. In their formulation, X is the data matrix of images and codewords, F is the codeword cluster matrix, S is the matrix associating codeword clusters and image classes, and G is the image class indicator matrix.

Li *et al.* [34] also use the matrix factorization technique, but for Collaborative Filtering. In their analysis, X is the ratings matrix for users and items, F contains item clustering information, S is the matrix associating item clusters and user clusters, and G contains user clustering information. It should be noted that Li *et al.* [34] were the first of [34], [33], and [32] to use this technique for transfer learning.

Parameter-based methods have a wide range of techniques and applications. For image classification tasks, [27] or [33] might be the best choice. For document classification, maybe [32] is the best choice. Many methods give very general solutions that can be applied to almost any traditional machine learning problem.

3.5 Deep Transfer Learning

Deep transfer learning is a relatively new topic in the field of transfer learning. With a goal analogous to feature-based methods, these methods attempt to align the feature representation that are found in the internal activations of the network for the Target and Source Domains. Even though each method in the following section can likely be categorized as a feature-based method or a parameter-based method, due to the high interest in the field, and for sake of continuity, we chose to include a separate section on this sub-field.

Deep methods can range from very simple techniques to the creation of much more involved architectures. Oquab *et al.* [35] use one such simple technique for object detection. They take the first 5 convolutional layers and the first 2 fully connected layers of a network pre-trained on ImageNet [36] and add two new fully connected layers, with a classification head, to the end of the network. This model is trained on the Target task using a sliding window object detection technique. The layers taken from the pre-trained network are fixed during training and only the new layers are optimized. In their experiments, ImageNet [36] is the Source Domain and Pascal VOC [37] is the Target Domain. We can see the structure of this transfer in Figure 3.

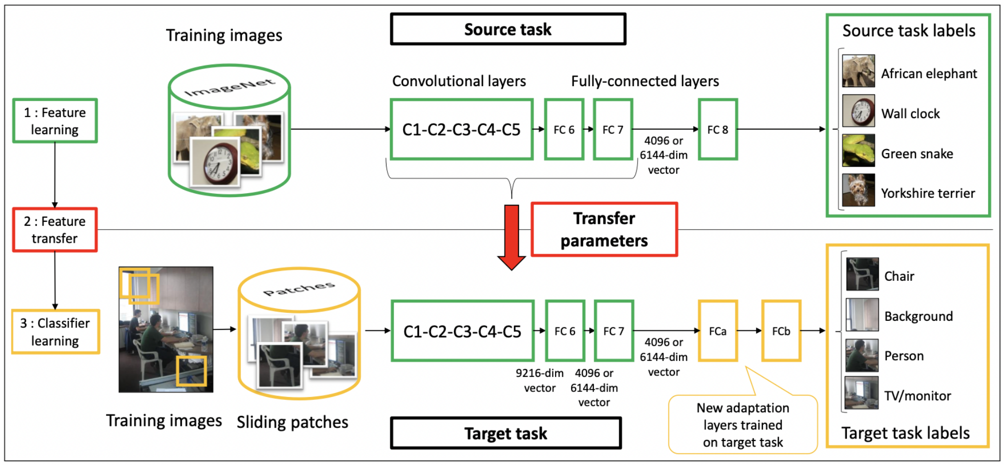


Figure 3: Transferring parameters of a CNN. Taken from [35]

Here the authors are basically training a small network on images using the internal activation of ImageNet as its input feature representation. This method provides a perfect example of how deep transfer learning is a combination of parameter and feature-based methods. We are directly taking parameters from a network trained on the Source, but in doing so, we are taking the feature representation from the network. This duality justifies the addition of a separate deep learning section to avoid confusion.

Long *et al.* [38] introduce a similar, but more involved method called Deep Adaptation Networks (DAN). DAN is a transfer learning network for image classification that can work with unlabeled Target data. The authors first take AlexNet, which is trained on ImageNet, and extract the first 5 convolutional layers. The first 3 are frozen during training while the last 2 are fine-tuned. After the last convolutional layer, they add 2 separate channels for each Domain, each with 3 fully connected layers and a softmax output over class labels. At each hidden layer of different channels, the activation's distributions are compared using a squared version of the multiple kernel variant of MMD (MK-MMD) where the kernel, k, is a convex combination of m PSD kernels, a technique very similar to that of [10]. While this statistic, $d_k^2(P_S, P_T)$, can be formulated as

$$d_k^2(P_S, P_T) = \mathbb{E}[k(x_i^S, x_j^S)] + \mathbb{E}[k(x_i^T, x_j^T)] - 2\mathbb{E}[k(x^S, x^T)],$$

estimating the expectation by using all of the samples has a time complexity of $O(n^2)$ and thus the authors use the $O(n)$ approximation,

$$d_k^2(P_S, P_T) = \frac{2}{n_S} \sum_{i=1}^{n_S/2} \Big[k(x_{2i-1}^S, x_{2i}^S) + k(x_{2i-1}^T, x_{2i}^T) - k(x_{2i-1}^S, x_{2i}^T) - k(x_{2i}^S, x_{2i-1}^T) \Big],$$

which works well with gradient descent. The authors try to minimize this term for all three layers in the channels and attempt to minimize the cross-entropy loss of the labeled Source data. Thus their objective function is

$$\min_{\theta} \quad \frac{1}{n_S} \sum_{i=1}^{n_S} J(f_\theta(x_i^S), y_i^S) + \lambda \sum_{\ell=6}^{8} d_k^2(h_\ell^S, h_\ell^T),$$

where $f_\theta(\cdot)$ is the network with parameters θ, $J(\cdot)$ is cross-entropy loss, and h_ℓ^D is the activations of the ℓ^{th} hidden layer for the channel corresponding to Domain D. The DAN architecture is depicted in Figure 4.

By ensuring that the distribution of activations are the same for the Target and the Source, we can expect that the output of the Target channel will be a decent classifier, even though it hasn't seen any labeled training examples. However, this technique can be trivially extended to the addition of labeled Target data, simply by adding a risk term for the Target Domain to the loss function.

Zhu *et al.* [39] build off of the work of Long *et al.* [38], extend the method to multiple Sources, and generalize the framework, calling it Multiple Feature Spaces Adaptation Network (MFSAN). Unlike [38] the authors build their network from scratch. The network starts with feature extractor F which

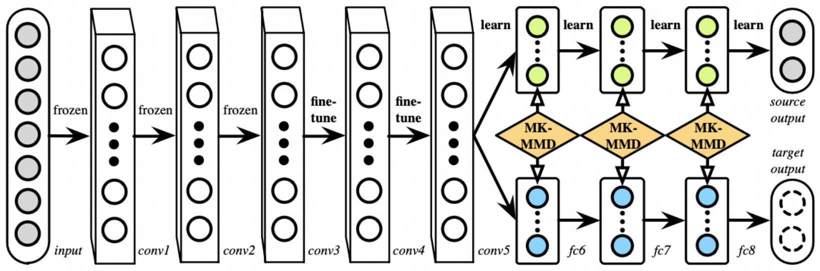


Figure 4: The DAN architecture. Taken from [38]

all data passes through. Then, the network separates into N channels, one for each Source Domain. Each channel starts with a Domain-specific feature extractor, H_s, which is followed by a Domain-specific classifier, C_s. While data from each Source only travels through its particular channel, the Target data travels through all channels. During training the MMD^2 of the outputs of each H_s for the Target and Source data are minimized. Additionally, the cross-entropy loss of each Source prediction is minimized. Finally, the difference in the prediction for Target examples from each Source channel is minimized. Thus, we have a 3 part objective function

$$\min_{\theta} \sum_{s=1}^{N} \sum_{i=1}^{n_s} J(f_\theta(x_i^s), y_i^s) + \lambda \frac{1}{N} \sum_{s=1}^{N} MMD^2(H_s(F(X^s)), H_s(F(X^T)))$$
$$+\gamma \frac{2}{N \cdot (N-1)} \sum_{s=1}^{N-1} \sum_{i=s+1}^{N} \mathbb{E}_{x \sim P_T}\Big[\big|C_i(H_i(F(x))) - C_s(H_s(F(x)))\big|\Big].$$

After training, the network can be used on the Target Domain by passing the Target data through all channels and then averaging the predictions for each channel to get a final label prediction. The MFSAN network can be seen in Figure 5.

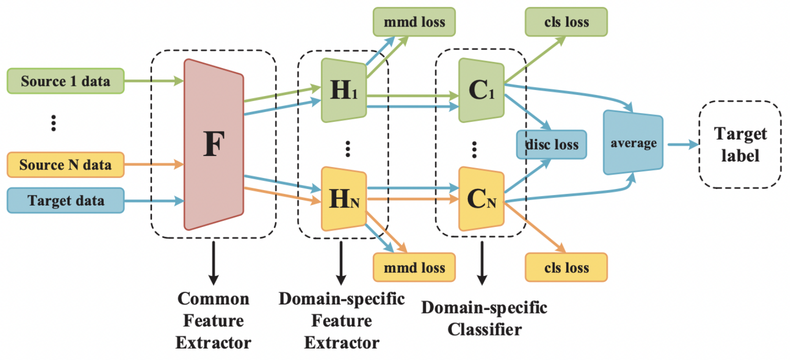


Figure 5: The MFSAN architecture. Taken from [39]

Zhuang *et al.* [40] develop a method for knowledge transfer using Deep Autoencoders. Given labeled Source data with C classes and unlabeled Target data, they construct an Autoencoder to work on both Domains. The input, x is passes through 2 hidden layers, parameterized by (W_1, b_1) and (W_2, b_2) respectively, where $\xi = f(W_1 x + b_1)$ and $\hat{y} = f(W_2 \xi + b_2)$. Here, $\hat{y}$, of size C, represents the softmax class label predictions. Then, $\hat{y}$ is passed through two more hidden layers, parameterized by (W_3, b_3) and (W_4, b_4) respectively, where $\hat{\xi} = f(W_3 \hat{y} + b_3)$ and $\hat{x} = f(W_4 \hat{\xi} + b_4)$. Here, $\hat{x}$ is the reconstruction of the input features. Their loss is then

$$\mathcal{J} = J_r(x_i, \hat{x}_i) + \alpha \Gamma(\xi^S, \xi^T) + \beta L(y_i^S, \hat{y}_i^S) + \gamma \Omega(W, b).$$

The first term is the reconstruction error for all data points, the second term is the KL-Divergence between the activations after the second hidden layer for each Domain, the third term is the classification error for the Source Domain, and the last term is a parameter regularization term. Because the internal activations are encouraged to be similar between Domains, the classification output for Target examples should have relatively high accuracy.

While previous examples minimize the distance between activation distributions explicitly, using MMD or KL Divergence, many methods do so implicitly. One example is Glorot *et al.* [41], who also use Autoencoders for knowledge transfer, but create a technique tailored to sentiment classification. First, they train Stacked Denoising Autoencoders (SDAs) on all Domains. Denoising Autoencoders take an input x that has entries randomly masked, resulting in $\tilde{x}$, which is then fed into the Autoencoder. The Autoencoder reconstructs the output as $\hat{x}$, which is then compared to the original input x. These can be stacked to form a chain of Denoising Autoencoders where the output for each network in the chain is the input to the next network. This implicitly creates a similar feature representation for all Domains inside of the SDA. A linear classifier can then be trained on the inner activations of the SDA using only the Source data.

Another way to implicitly force the activations to be distributed the same is through adversarial training. Ganin *et al.* [42] and later Ganin *et al.* [43] use this technique to align the Domains inside of the network, calling their method Domain-Adversarial Neural Network (DANN). Their network starts with a feature extractor, G_f, which both Domains pass through. After that, the network splits into two channels. The first channel, G_y, predicts the class label, while the second channel,G_d predicts the Domain of the data. G_d is trained adversarially, which means the channel tries to minimize the classification error. When gradients are passed from the channel, back to the G_f, they are reversed and scaled by λ. Thus, while G_d attempts to tell the Domains apart, G_f is incentivized to make its output identically distributed across Domains. Therefore, the loss of the entire network can be written as

$$\mathcal{L}(\theta_f, \theta_y, \theta_d) = \frac{1}{n_S}\sum_{i=1}^{n_s}\mathcal{L}(\theta_f, \theta_y, x_i^S, y_i^S) - \lambda\left[\frac{1}{n_S}\sum_{i=1}^{n_S}\mathcal{L}(\theta_f, \theta_d, x_i^S, D_S) + \frac{1}{n_T}\sum_{i=1}^{n_T}\mathcal{L}(\theta_f, \theta_d, x_i^T, D_T)\right],$$

where the first term is the label classification loss on the Source Domain and the second 2 terms are the Domain classification loss on each Domain. This optimization of this loss function can be written as

$$\min_{\theta_f, \theta_y}\left[\max_{\theta_d}\left[\mathcal{L}(\theta_f, \theta_y, \theta_d)\right]\right].$$

The architecture for this model can be seen in Figure 6.

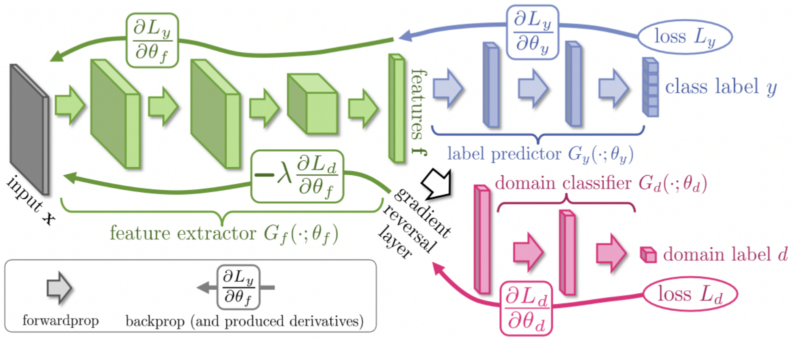

$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features f
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
domain classifier $G_d(\cdot;\theta_d)$
input x
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal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

Figure 6: The DANN architecture. Taken from [42]

You *et al.* [44] create a very general framework called Universal Domain Adaptation (UDA). They attempt to extend the adversarial network, to predict when an example in the Target set is from a class that is not seen in the Source. They let $\mathcal{C}_S$ be the classes in the Source Domain and $\mathcal{C}_T$ be the classes in the Target Domain, with $\mathcal{C} = \mathcal{C}_S \bigcap \mathcal{C}_T$ as the labels common to both Domains. Then, their model consists of 4 parts: a feature extractor F, an adversarial Domain discriminator D, a non-adversarial Domain discriminator D' and a label classifier G. The feature extractor outputs the embedding $z = F(x)$ which is passed to the other three parts which have outputs: $\hat{y} = G(z)$, $\hat{d}' = D'(z)$, and $\hat{d} = D(z)$, where $\hat{y}$ is the predicted label, $\hat{d}'$ is a measure of Domain similarity, and $\hat{d}$ is the Domain prediction. The composed network, called Universal Adaptation Network (UAN), can be seen in Figure 7.

The error for the various parts can be written as

$$\begin{aligned} E_G &= \mathbb{E}_{(x,y)\sim P_S} L(y, G(F(x))) \\ E_{D'} &= -\mathbb{E}_{x\sim P_S}\log\Big(D'(F(x))\Big) - \mathbb{E}_{x\sim P_T}\log\Big(1 - D'(F(x)\Big) \\ E_D &= -\mathbb{E}_{x\sim P_S} w^S(x)\log\Big(D(F(x))\Big) - \mathbb{E}_{x\sim P_T} w^T(x)\log\Big(1 - D(F(x)\Big) \end{aligned}$$

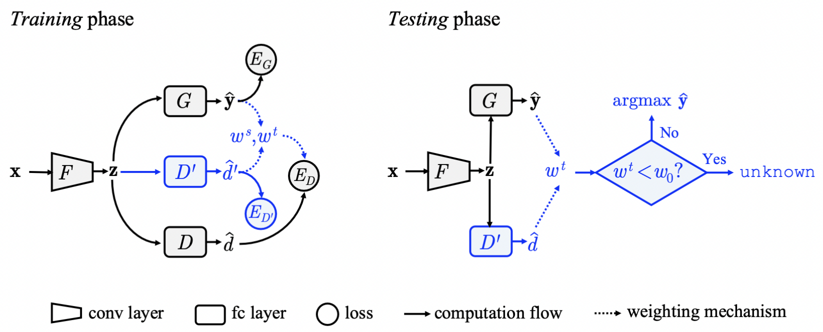


Figure 7: UAN architecture for Universal Domain Adaptation (UDA). Taken from [44]

and the training of the model can be written as

$$\max_{D} \min_{F,G} E_G - \lambda E_D$$

$$\max_{D'} E_{D'}.$$

The predicted output $\hat{y}$ for a given Target example is then checked for confidence over the threshold w_0 such that

$$y(x^T) = \begin{cases} \arg\max(\hat{y}) & w^T \geq w_0 \\ \text{unknown} & w^T < w_0 \end{cases}$$

Where $w^S(x)$ and $w^T(x)$ are the confidence weight for each Domain, computed as

$$w^S(x) = \frac{H(\hat{y})}{\log(|\mathcal{C}_S|)} - \hat{d}'(x)$$
$$w^T(x) = \hat{d}'(x) - \frac{H(\hat{y})}{\log(|\mathcal{C}_S|)},$$

where $H(\hat{y})$ is the entropy of $\hat{y}$. Thus, to make a prediction of a Source class for a Target example, the model requires a certain degree of confidence, otherwise, it will return unknown. The main difference between UDA and other adversarial networks for transfer learning is that UDA includes the non-adversarial Domain discriminator to determine how close the Target example is to the Source Domain.

We choose to leave out a performance comparison of deep learning techniques as there is a lack of consistency in the comparison of the methods. Most notably, many of the papers compare their method to various feature-based approaches for the Office data set [26], but use slightly different training techniques for the compared methods such that the accuracy for the same model on the same task differs a substantial amount across papers. For example, for the $D \rightarrow W$ setting for the Office data set, [42] compare their method to GFK where it has 69.1% accuracy while [38] compare their method to GFK where it has 49.7% accuracy. Thus individually reported performances cannot be accurately aggregated.

3.6 Relational Methods

Relational transfer learning is the rarest type of transfer learning. These methods involve finding some relationship between aspects of the Source and Target Domain and using this relationship to transfer knowledge. However, this means that each method is usually tailored toward a certain field and cannot be formed into a generalized framework. Due to the scarcity of relational-based literature, we only include one such example, from Li *et al.* [45]

Their method involves using bootstrapping to extract sentiment and topic lexicons from documents. They assume a setting where each word in the Source document, and some words in the Target document, are labeled as one of {Topic, Sentiment, Neither}. They iteratively update a bipartite graph between Topic and Sentiment words where the weights are based on how associated the words are in the Target Domain. The idea behind this is that if Sentiment words are similar across Domains, then the same Sentiment word will be similarly connected to topic words. An example of such a graph can be seen in Figure 8.

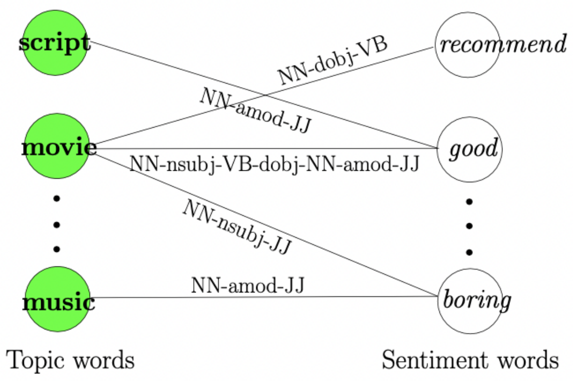


Figure 8: Example of the bipartite graph constructed using RAP. Taken from [45]

The authors use a modified version of TrAdaBoost called Relational Adaptive bootstrapping (RAP) to construct the graph. During training, the unlabeled Target data is continuously selected to be pseudo-labeled and added to the training set. The weights of the graph are also updated based on a scoring function similar to the one used in TrAdaBoost. The algorithm returns the set of Topic and Sentiment words in the Target Domain

4 Negative Transfer

In transfer learning, we assume that the Domains are sufficiently similar to enhance the accuracy of a model by leveraging the Source data. However, this is not always the case. Negative Transfer occurs when knowledge transfer from a Source Domain adversely affects models in the Target Domain.

Research in Negative Transfer tries to explain when this might happen and how to stop it.

Rosenstein *et al.* [46] develop an extremely simple, "transfer aware" version of Naive Bayes, that can avoid negative transfer, even with a limited number of labeled Target instances. They train two models simultaneously, one on the Source, with parameters θ_S, and one on the Target, with parameters θ_T. They maximize the posterior of the two models where a hyperprior is paced over $(\theta_S - \theta_T)$ with learnable parameters to control the tightness of the distribution. Since the authors optimize over the hyperprior parameters as well, the model can choose the optimal tightness of the prior, given the data. This roughly suggests that the model can choose whether to transfer or not. The authors test this method on Domains that are known to be dissimilar and find that while the method always performs worse than training on just Target data, as the number of labeled Target examples increases, the method asymptotes to the accuracy of the Target model.

Ge *et al.* [47] introduce a framework for multi-Source Transfer that seeks to avoid Negative Transfer. The authors assume they have limited labeled Target data and an abundance of unlabeled Target data. They first use spectral clustering to form c clusters by minimizing

$$\min_f \frac{1}{2}\sum_{i,j=1}^{n_T} W_{ij}\left(\frac{f_i}{\sqrt{D_{ii}}} - \frac{f_j}{\sqrt{D_{jj}}}\right)^2,$$

where W is the similarity matrix, D_{ii} is the sum of the i^{th} row of W and f_i is the cluster id. Then, using the limited labeled Target data, we can pseudo-label the Target data by minimizing

$$\min_F \frac{1}{2}\left(\sum_{i,j=1}^{n_T} W_{ij}\left\|\frac{F_i}{\sqrt{D_{ii}}} - \frac{F_j}{\sqrt{D_{jj}}}\right\|^2 + \mu\sum_{i=1}^{n^T}\left\|F_i - H_i\right\|^2\right),$$

where $F_i \in \mathbb{R}^c$ denotes pseudo-labeled class membership and $H_i \in \mathbb{R}^c$ denotes labeled data class membership. We can then create a Source classifier using the equation

$$\hat{h}_i = \sum_{s=1}^{k} f_i^s \cdot w_{C_i,s},$$

where f_i^s is a classifier trained on Source s and $w_{C_i,s}$ is the percent of label matches between the pseudo-labels and the labels predicted by f_i^S for cluster C_i. Then our final objective function to obtain a Target classifier f^T is

$$\min_{f^T} \frac{1}{n_T^l}\sum_{i=1}^{n_T^l}(f_i^T - H_i)^2 + \gamma\|f^T\|_K^2 + \frac{\beta}{2}\sum_{j=n_T^l+1}^{n_T}\|f_j^T - \hat{h}_j\|^2.$$

Therefore, we constrain the Target Classifier to be close to the labeled Target data (term 1), to not be too complex (term 2), and to be close to the combined Source label for that particular cluster (term 3). The researchers' experiments show that this technique can pick out relevant Sources from the set of available Sources and generally avoid Negative Transfer.

Seah *et al.* [48] develop an iterative algorithm to combat negative transfer for the multi-Source setting, with k Sources. They assume no labeled data in the Target Domain (which is the $(k+1)^{th}$ Domain in their notation) is available. Their algorithm is as follows.

1. Train a Predictive Distribution Matching (PDM) regularized classifier on all of the labeled (and pseudo-labeled) data. The PDM regularization term takes the form

$$\frac{1}{(\sum_{i=1}^{k+1} n_i)^2} \sum_{r,d=1}^{k+1} \sum_{i=1}^{n_r} \sum_{j=1, j\neq i}^{n_d} \left(f(x_i^r) - f(x_j^d) \right)^2 W_{ij}^{rd} \cdot \mathbb{1}[r \neq d]$$

   Where W_{ij}^{rd} is the similarity of $x_i \in D_r$ and $x_j \in D_d$.

2. Pseudo-label the Target data using $f(\cdot)$, with the highest confidence pseudo-labeled data for each class being added to the training set in (1).

3. Train a softmax classifier on the set of pseudo-labeled Target data in the training set for (1) and use the output to get the confidence of each Source data's true label in the training set. The lowest confidence examples for each class are removed. Check for convergence (when the lowest confidence pseudo-label falls below a certain level). If convergence has been met, go to (4), else, go to (1).

4. Train a Target classifier on all of the labeled data (and pseudo-labeled) data.

5. Predict the labels of all other Target data.

Because this method removes low-confidence Source data at each iteration, it has to potential to remove Source data that might cause the model to perform worse on the Target Domain. This means the technique can avoid Negative Transfer in some cases. However, since we don't have access to labeled Target data in this scenario, we can't guarantee that this method will always combat Negative Transfer. That being said, in their experiments, the researchers demonstrate that this method usually outperforms other transfer learning methods (that don't focus on Negative Transfer) especially when there is a large discrepancy between the predictive distributions of the Source and Target Domains (i.e the Domains are very different).

5 Applications

While transfer learning is a general tool that can be applied to almost any machine learning scenario, there is a large amount of literature on applications in Natural Language Processing (NLP), collaborative filtering, and computer vision.

5.1 NLP

As demonstrated in the previous sections, transfer learning has many applications in NLP since in Natural Language, words generally keep the same meaning across Domains and thus there is a high potential for knowledge transfer. The most common NLP setting that transfer learning is used in is sentiment analysis. We have already seen a Matrix Tri-Factorization technique from Zhuang *et al.* [32], a Relational Bootstrapping method from Li *et al.* [45], and a deep method from Glorot *et al.* [41], however, there are many more transfer learning works on sentiment analysis, including the following.

Dai *et al.* [49] use a word-document co-clustering approach to classify documents. They iteratively co-cluster words and documents in the Target Domain based on a mutual information criteria. They then use the word clustering to extract label information from the Source Domain documents.

Pan *et al.* [50] use a different technique for document classification. They first determine Domain independent words by the frequency of each word in each Domain. Then they construct a bipartite graph between Domain independent and Domain dependent words. Finally, they use spectral clustering on the graph to create additional features that a classifier can be trained on.

Xia *et al.* [51] propose a 2 stage method for sentiment classification. In the first stage, they perform PCA on the Target Domain, extracting the top k features. They then project the Source data into the feature space and remove the Source data that is far away from the Target distribution in this new feature space. In the second stage, they train a sentiment classifier on the chosen Source data for each part of speech and use an ensemble of these classifiers to construct their final classifier, which should be effective in the Target Domain.

5.2 Collaborative Filtering

Transfer learning is especially useful when training collaborative filtering models. Because user-item ratings matrices are usually sparse, transfer learning can help fill the gaps in the matrix using Source data. We saw how Li *et al.* [34] tackled this problem in the feature-based section, but there are many other common techniques in this area, including Pan *et al.* [52] who also create a method for filling sparse user-ratings matrices. They take uncertainty ranges for each user-item pair as a prior and fit a model that predicts the sparse entries in the matrix using user and item latent features, with the uncertainty as a regularizer.

5.3 Object Detection and Image Classification

Computer vision is a field that, for similar reasons to Natural Language, offers many opportunities for applying transfer learning. Images work well for knowledge transfer because there exist large amounts of unlabeled data, and most images share similar representations in some unknown feature space. Many of the aforementioned methods were specifically designed for image classification or object detection. These include the prototype space technique in Quattoni *et al.* [17], the Matrix Tri-Factorization method in Wang *et al.* [33], the deep learning approach in Oquab *et al.* [35], and the probabilistic method of Stark *et al.* [27]. Additionally, some other methods in this field are detailed below.

Duan *et al.* [53] develop a method for event recognition in videos, using image classifiers trained on Source Domains. The Target classifier is a weighted combination of these Source classifiers, averaged over all frames of the video, and a 3D (space and time) feature classifier. They create an optimization problem that selects a subset of the Source classifiers and forces the Target classifier to have similar predictions to this subset.

Yu *et al.* [54] develop an image classification method that uses image features to predict image attributes. They first train a generative model on the Source Domain of images, which can then be used to transfer knowledge in two distinct ways. First, given the Target attributes, they can generate image features and add the synthetic examples to the Target training data set. Second, they can replace the flat priors on the model's parameters with priors centered on the Source model's parameters.

Di *et al.* [55] introduce a knowledge transfer method for understanding traffic scenes in different weather conditions. Given an annotated traffic scene in one weather condition, they hope to transfer the annotations to the same scene in a different weather condition. The images from all Domains are passed through a CNN to extract features. These features are further reduced using Partial Least Squares (PLS) regression and then aligned using metric learning, such that the Domains all have similar feature representations. Finally, they use SIFT Flow to find the corresponding image of the same scene in different weather and transfer the annotations to the Target Domain.

Maqsood *et al.* [56] use a transfer learning technique to detect different stages of Alzheimer's Disease. They take the convolutional layers from AlexNet, pre-trained on ImageNet, add new fully connected layers and a classification head, and then train the model on MRI data.

6 Future Work

While the field of transfer learning has been around for many years, the majority of work has been related to finding new algorithms that combine some Domain assumptions with a machine learning technique. In the early 2000s, these techniques were fairly simple, but have since evolved into much more complex algorithms in an attempt to create a novel approach. More recently, the field of deep learning has taken off and many researchers have seized the opportunity to combine transfer learning with these black box models. However, during this time, there has been a lack of adequate research into the theory of transfer learning. To this day there are many unanswered questions in the field. These include: “How can we control or view what information is being transferred across Domains?”, “When and why do these techniques yield Negative Transfer?”, and “What are the theoretical limits to the amount of knowledge we can transfer from a Domain?”. Some feel that answering, or even simply exploring these types of questions would do much to advance the field as a whole. Highlighted below are a few papers that start to explore these questions.

Neyshabur *et al.* [57] deliver an in-depth analysis of the effect of pre-training on knowledge transfer. In their experiments, they use ImageNet as their pre-training (Source) data set and DomainNet [58] and CheXpert [59] their fine-tuning (Target) data sets. The first experiment they run is to randomly shuffle pixel blocks in the fine-tuning data set for various block sizes. A block size of 1 implies all pixels are randomly shuffled, while a block size of 3 means 3x3 blocks of pixels are shifted around and a block size of 224 (the image size) means the image stays fully intact. This essentially keeps low-level image statistics intact, but visually moves the Source and Target Domains very far away from each other.

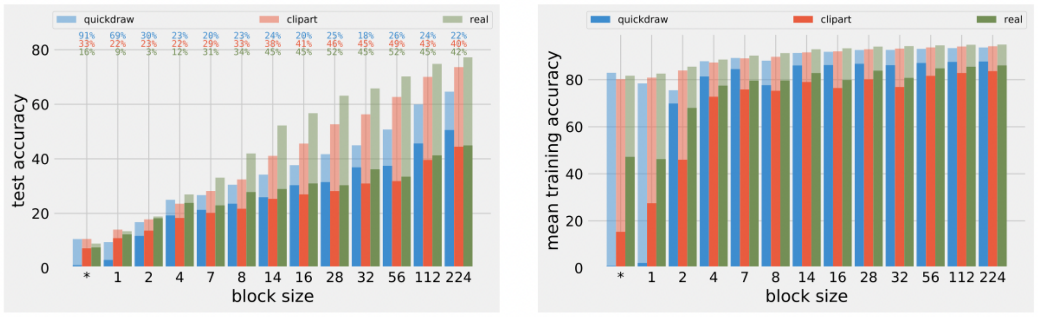
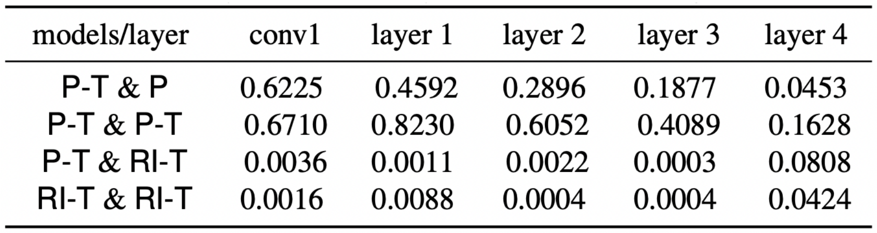


Figure 9: The effect of block sizes on final performance. The semi-transparent bars correspond to pre-trained networks while the solid bars correspond to randomly initialized networks. Taken from [57]

As seen in Figure 9, the left plot shows diminishing returns from pre-training when the Target and Source Domains become increasingly different. However, for training speed, the pre-trained model consistently outperforms the randomly initialized models. Both of these imply that visual features are very important to transfer learning, but low-level statistics also come into play at some point.

The second experiment they perform is to compare the internal feature similarity of 3 models: the pre-trained model, the pre-trained and fine-tuned model, and the randomly initialized and then trained model. They use Centered Kernel Alignment (CKA) at various layers of the network to compare similarity.



models/layer	conv1	layer 1	layer 2	layer 3	layer 4
P-T & P	0.6225	0.4592	0.2896	0.1877	0.0453
P-T & P-T	0.6710	0.8230	0.6052	0.4089	0.1628
P-T & RI-T	0.0036	0.0011	0.0022	0.0003	0.0808
RI-T & RI-T	0.0016	0.0088	0.0004	0.0004	0.0424

Figure 10: Feature similarity for different layers of ResNet-50. **P**: network just pre-trained on the Source. **P-T**: network pre-trained on the Source and fine-tuned on the Target. **RI-T**: network randomly initialized and trained on the Target. Taken from [57]

In Figure 10, we can see that fine-tuned models are relatively close to each other and to their corresponding pre-trained network. This similarity is strongest in the initial layers and weakens further into the network. On the other hand, randomly initialized networks are dissimilar from each other and the fine-tuned network at all depths in the architecture. These results might mean that early layers, which are assumed to learn low-level features of the data, are much more involved in transferring knowledge.

The third experiment the researchers run compares the loss basin of different models. By interpolating between model parameters from two different networks, and evaluating the loss of the new interpolated network, we can determine if different networks are located in the same loss basin. The results of this experiment show that fine-tuned models reside in the same loss basin, while randomly initialized (and then trained) models reside in separate loss basins, even when they are initialized with the same random weights. This implies that pre-training guides the final fine-tuned networks to the same local minimum. Thus, given a clever choice of a Source Domain, a researcher might be able to have some influence over where the Target model converges.

Zhang *et al.* [60] develop a new Domain comparison metric, Margin Disparity Discrepancy (MDD) for classifiers using scoring functions and margin loss. This new metric is defined by a supremum over a single hypothesis space which induces a minimax game when the metric is minimized. This allows the creation of an adversarial network architecture that fits perfectly with the theory of the loss function. The authors set up the notation by defining a scoring function $f : \mathcal{X} \to \mathbb{R}^{|y|}$, where the notation $f(x, y) \in \mathbb{R}$ refers to the score for the given y. A hypothesis is defined as $h_f : x \to \arg\max_{y \in \mathcal{Y}} f(x, y)$.

First, let

$$\text{disp}_D(h', h) = \mathbb{E}_D \mathbb{1}[h' \neq h]$$

be the Expected 0-1 disparity between two hypotheses $h', h \in \mathcal{H}$. Then, given a hypothesis space $\mathcal{H}$ and a hypothesis $h \in \mathcal{H}$, the Disparity Discrepancy induced by $h' \in \mathcal{H}$ between two Domains, D_S and D_T, is

$$d_{h,\mathcal{H}}(D_T, D_S) = \sup_{h' \in \mathcal{H}} \Big(\text{disp}_{D_S}(h', h) - \text{disp}_{D_T}(h', h) \Big).$$

Let f and f' be the scoring functions associated with the hypothesis h and h' respectively. We can then define the Margin Discrepancy as

$$\text{disp}^{\rho}_{D}(f', f) = \mathbb{E}_{x \sim D} \Phi \Big(\rho_{f'}(x, h_f) \Big),$$

where $\rho_{f'}(x, y) = \frac{1}{2} \Big(f'(x, y) - \max_{y' \neq \mathcal{Y}} f'(x, y') \Big)$ is the margin of hypothesis f' and Φ is defined such that

$$\Phi = \begin{cases} 0 & \rho \leq x \\ 1 - x/\rho & 0 \leq x \leq \rho \\ 1 & x \leq 0 \end{cases}.$$

We can then define Margin Disparity Discrepancy as

$$d_{f,\mathcal{F}}(D_T, D_S) = \sup_{f' \in \mathcal{F}} \Big(\text{disp}^{\rho}_{D_S}(f', f) - \text{disp}^{\rho}_{D_T}(f', f) \Big).$$

Given this metric, the authors introduce the architecture of a simple network, seen in Figure 11, to minimize the MDD and also train the classifier on Source labels

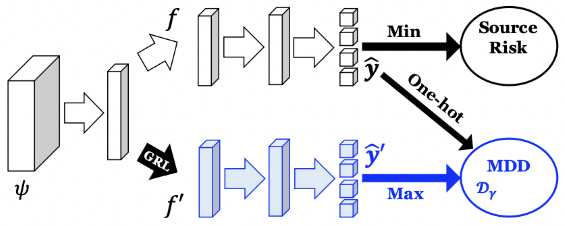


Figure 11: The proposed architecture to minimize MDD over the Source and Target Domains. The channel in blue is adversarially trained with GRL indicating a Gradient Reversal Layer. Taken from [60]

This work demonstrates how using theory to aid in the construction of models can work to the researcher's benefit. By creating a metric that has a minimax game built into the theory, the authors were able to trivially construct a network with a loss function that is optimized for this metric. This method performed very well against similar transfer learning techniques.

Zhao *et al.* [61] explore the idea that simply finding a feature space where the marginal distributions are the same does not guarantee positive transfer. They provide an example where a classifier could achieve 0 error when trained on either Domain, but would have an error of 1 (the maximum error) when the Domains were combined in a feature space. The authors further provide upper and lower bounds for the Target classification error,

$$\epsilon_T(h) \leq \epsilon_S(h) + d_{\tilde{\mathcal{H}}}(D_S, D_T) + \min\Big(\mathbb{E}_{D_S}\Big[\big|f_S - f_T\big|\Big], \mathbb{E}_{D_T}\Big[\big|f_S - f_T\big|\Big]\Big),$$

where $d_{\mathcal{H}}$ is $\mathcal{H}$ divergence and $\tilde{\mathcal{H}} = \{\mathrm{sgn}(|h(x) - h'(x)| - t)|h, h' \in (H), t \in [0,1]\}$. In this bound, the first term corresponds to the Source error, the second term corresponds to the differences in the marginal distributions, and the third term corresponds to the differences in the labeling functions for each Domain. The authors also provide an upper bound

$$\epsilon_T(h \circ g) \geq -\epsilon_S(h \circ g) + \frac{1}{2}\left(\sqrt{D_{JS}(D_S^Y \parallel D_T^Y)} - \sqrt{D_{JS}(D_S^Z \parallel D_T^Z)}\right)^2,$$

where D_{JS} is Jensen-Shannon distance and g is the mapping to a feature space Z such that $X \xrightarrow{g} Z \xrightarrow{h} Y$. Additionally, D_i^Z and D_i^Y are the induced distributions in Z and Y space respectively for Domain i. In this upper bound, the first term is the Source error using the intermediary feature space and the second two terms are the differences in the label distributions minus the differences in the embedding distributions.

While many assumptions and the majority of the associated proofs are left out of the analysis above, Zhao *et al.* were successful in finding an upper and lower bound for knowledge transfer. Additionally, they derived the bounds in such a way that was interpretable, and shed light on what conditions Domain adaptation needs in order to be successful.

7 Conclusion

In this paper, we have defined the transfer learning setting, classified the various methods, and introduced many techniques used in the field. For instanced based methods, we derived the desired weight ratio needed for instance weighting and explored various estimation methods. For feature-based methods, we went over some of the common assumptions in the field and examined how various researchers used these assumptions to develop novel representations for the data. For parameter-based methods, we investigated the ability of model parameters to hold knowledge. For deep methods, we surveyed recent works that took traditional inspiration from feature and parameter-based methods as well as works that took the field in a new direction with adversarial training. For relational-based methods, we dove into the work of Li *et al.* [45] and how they used domain knowledge to their benefit.

From the distribution of dates for this paper's citations, we can see that homogeneous transfer learning is a relatively old technique, however, the abundance of landmark transfer learning papers from the past 8 years shows that this field has much more to offer machine learning. The explosion of deep learning and the new focus on Big Data will only drive the evolution of transfer learning to new heights as the demand for new sources of data increases.

While the application of transfer learning to various machine learning settings has a bright future, there needs to be more future work in the theoretical aspects of the field. Deriving bounds on transferability, working towards the explainability of knowledge transfer, and the prevention and analysis of negative transfer are essential to the efficacy of the field in the years to come.

References

[1] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

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

[3] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola, "Correcting sample selection bias by unlabeled data," *Advances in neural information processing systems*, vol. 19, 2006.

[4] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning.(2007), 193–200," in *Proceedings of the 24th international conference on Machine learning*, 2007.

[5] Y. Yao and G. Doretto, "Boosting for transfer learning with multiple sources," in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 1855–1862, IEEE, 2010.

[6] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. B nau, and M. Kawanabe, "Direct importance estimation for covariate shift adaptation," *Annals of the Institute of Statistical Mathematics*, vol. 60, no. 4, pp. 699–746, 2008.

[7] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in nlp," ACL, 2007.

[8] Q. Sun, R. Chattopadhyay, S. Panchanathan, and J. Ye, "A two-stage weighting framework for multi-source domain adaptation," *Advances in neural information processing systems*, vol. 24, 2011.

[9] S. J. Pan, J. T. Kwok, Q. Yang, *et al.*, "Transfer learning via dimensionality reduction.," in *AAAI*, vol. 8, pp. 677–682, 2008.

[10] L. Duan, I. W. Tsang, and D. Xu, "Domain transfer multiple kernel learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 465–479, 2012.

[11] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, "Unsupervised domain adaptation by domain invariant projection," in *Proceedings of the IEEE international conference on computer vision*, pp. 769–776, 2013.

[12] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE transactions on neural networks*, vol. 22, no. 2, pp. 199–210, 2010.

[13] 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*, pp. 2200–2207, 2013.

[14] M. Long, J. Wang, G. Ding, S. J. Pan, and S. Y. Philip, "Adaptation regularization: A general framework for transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1076–1089, 2013.

[15] S. Si, D. Tao, and B. Geng, "Bregman divergence-based regularization for transfer subspace learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 7, pp. 929–942, 2009.

[16] H. Daumé III, "Frustratingly easy domain adaptation," *arXiv preprint arXiv:0907.1815*, 2009.

[17] A. Quattoni, M. Collins, and T. Darrell, "Transfer learning for image classification with sparse prototype representations," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[18] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proceedings of the 2006 conference on empirical methods in natural language processing*, pp. 120–128, 2006.

[19] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 2066–2073, IEEE, 2012.

[20] Y. Shi and F. Sha, "Information-theoretical learning of discriminative clusters for unsupervised domain adaptation," *arXiv preprint arXiv:1206.6438*, 2012.

[21] 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)*, pp. 1129–1134, IEEE, 2017.

[22] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

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

[24] S. A. Nene, S. K. Nayar, H. Murase, *et al.*, "Columbia object image library (coil-20)," 1996.

[25] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[26] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, pp. 213–226, Springer, 2010.

[27] M. Stark, M. Goesele, and B. Schiele, "A shape-based object class model for knowledge transfer," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 373–380, IEEE, 2009.

[28] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A dasvm classification technique and a circular validation strategy," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 5, pp. 770–787, 2009.

[29] T. Tommasi and B. Caputo, "The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories," in *Proceedings of the British Machine Vision Conference*, no. CONF, pp. 80–1, 2009.

[30] T. Tommasi, F. Orabona, and B. Caputo, "Safety in numbers: Learning categories from few examples with multi model knowledge transfer," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3081–3088, IEEE, 2010.

[31] R. Chattopadhyay, Q. Sun, W. Fan, I. Davidson, S. Panchanathan, and J. Ye, "Multisource domain adaptation and its application to early detection of fatigue," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 4, pp. 1–26, 2012.

[32] F. Zhuang, P. Luo, H. Xiong, Q. He, Y. Xiong, and Z. Shi, "Exploiting associations between word clusters and document classes for cross-domain text categorization," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 4, no. 1, pp. 100–114, 2011.

[33] H. Wang, F. Nie, H. Huang, and C. Ding, "Dyadic transfer learning for cross-domain image classification," in *2011 International conference on computer vision*, pp. 551–556, IEEE, 2011.

[34] B. Li, Q. Yang, and X. Xue, "Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction," in *Twenty-First international joint conference on artificial intelligence*, 2009.

[35] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724, 2014.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[37] Z. Dong, K. Xu, Y. Yang, H. Bao, W. Xu, and R. W. Lau, "Location-aware single image reflection removal," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5017–5026, 2021.

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

[39] Y. Zhu, F. Zhuang, and D. Wang, "Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 5989–5996, 2019.

[40] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

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

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

[43] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[44] K. You, M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Universal domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2720–2729, 2019.

[45] F. Li, S. J. Pan, O. Jin, Q. Yang, and X. Zhu, "Cross-domain co-extraction of sentiment and topic lexicons," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 410–419, 2012.

[46] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, "To transfer or not to transfer," in *NIPS 2005 workshop on transfer learning*, vol. 898, 2005.

[47] L. Ge, J. Gao, H. Ngo, K. Li, and A. Zhang, "On handling negative transfer and imbalanced distributions in multiple source transfer learning," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 7, no. 4, pp. 254–271, 2014.

[48] C.-W. Seah, Y.-S. Ong, and I. W. Tsang, "Combating negative transfer from predictive distribution differences," *IEEE transactions on cybernetics*, vol. 43, no. 4, pp. 1153–1165, 2012.

[49] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Co-clustering based classification for out-of-domain documents," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 210–219, 2007.

[50] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment," in *Proceedings of the 19th international conference on World wide web*, pp. 751–760, 2010.

[51] R. Xia, C. Zong, X. Hu, and E. Cambria, "Feature ensemble plus sample selection: domain adaptation for sentiment classification," *IEEE Intelligent Systems*, vol. 28, no. 3, pp. 10–18, 2013.

[52] W. Pan, E. Xiang, and Q. Yang, "Transfer learning in collaborative filtering with uncertain ratings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, pp. 662–668, 2012.

[53] L. Duan, D. Xu, and S.-F. Chang, "Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach," in *2012 IEEE Conference on computer vision and pattern recognition*, pp. 1338–1345, IEEE, 2012.

[54] X. Yu and Y. Aloimonos, "Attribute-based transfer learning for object categorization with zero/one training example," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part V 11*, pp. 127–140, Springer, 2010.

[55] S. Di, H. Zhang, C.-G. Li, X. Mei, D. Prokhorov, and H. Ling, "Cross-domain traffic scene understanding: A dense correspondence-based transfer learning approach," *IEEE transactions on intelligent transportation systems*, vol. 19, no. 3, pp. 745–757, 2017.

[56] M. Maqsood, F. Nazir, U. Khan, F. Aadil, H. Jamal, I. Mehmood, and O.-y. Song, "Transfer learning assisted classification and detection of alzheimer's disease stages using 3d mri scans," *Sensors*, vol. 19, no. 11, p. 2645, 2019.

[57] B. Neyshabur, H. Sedghi, and C. Zhang, "What is being transferred in transfer learning?," *Advances in neural information processing systems*, vol. 33, pp. 512–523, 2020.

[58] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1406–1415, 2019.

[59] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, *et al.*, "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 590–597, 2019.

[60] Y. Zhang, T. Liu, M. Long, and M. Jordan, "Bridging theory and algorithm for domain adaptation," in *International conference on machine learning*, pp. 7404–7413, PMLR, 2019.

[61] H. Zhao, R. T. Des Combes, K. Zhang, and G. Gordon, "On learning invariant representations for domain adaptation," in *International conference on machine learning*, pp. 7523–7532, PMLR, 2019.