Distant Domain Transfer Learning

Ben Tan,[*] Yu Zhang,[*] Sinno Jialin Pan,[**] Qiang Yang[*]
[*]Hong Kong University of Science and Technology, Hong Kong
[**]Nanyang Technological University, Singapore
{btan,zhangyu,qyang}@cse.ust.hk, sinnopan@ntu.edu.sg

Abstract

In this paper, we study a novel transfer learning problem termed Distant Domain Transfer Learning (DDTL). Different from existing transfer learning problems which assume that there is a close relation between the source domain and the target domain, in the DDTL problem, the target domain can be totally different from the source domain. For example, the source domain classifies face images but the target domain distinguishes plane images. Inspired by the cognitive process of human where two seemingly unrelated concepts can be connected by learning intermediate concepts gradually, we propose a Selective Learning Algorithm (SLA) to solve the DDTL problem with supervised autoencoder or supervised convolutional autoencoder as a base model for handling different types of inputs. Intuitively, the SLA algorithm selects usefully unlabeled data gradually from intermediate domains as a bridge to break the large distribution gap for transferring knowledge between two distant domains. Empirical studies on image classification problems demonstrate the effectiveness of the proposed algorithm, and on some tasks the improvement in terms of the classification accuracy is up to 17% over "non-transfer" methods.

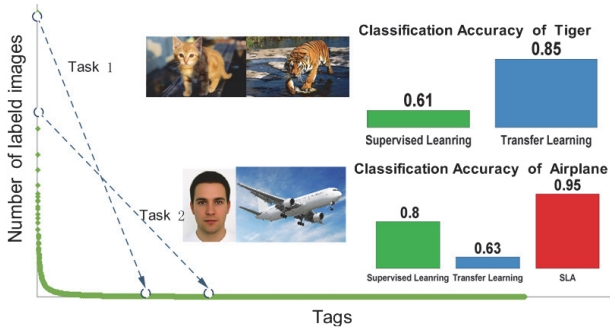


Figure 1: The tag distribution of the uploaded images at Qzone. In the first task, we transfer knowledge between cat and tiger images. A transfer learning algorithm achieves much better performance than some supervised learning algorithm. In the second task, we transfer knowledge between face and airplane images. The transfer learning algorithm fails as it achieves worse performance than the supervised learning algorithm. When applying our proposed SLA algorithm, however, we find that our model achieves much better performance.

Introduction

Transfer Learning, which borrows knowledge from a source domain to enhance the learning ability in a target domain, has received much attention recently and has been demonstrated to be effective in many applications. An essential requirement for successful knowledge transfer is that the source domain and the target domain should be closely related. This relation can be in the form of related instances, features or models, and measured by the KL-divergence or $\mathcal{A}$-distnace (Blitzer et al. 2008). For two distant domains where no direct relation can be found, transferring knowledge between them forcibly will not work. In the worst case, it could lead to even worse performance than 'non-transfer' algorithms in the target domain, which is the 'negative transfer' phenomena (Rosenstein et al. 2005; Pan and Yang 2010). For example, online photo sharing communities, such as Flickr and Qzone[1], generate vast amount of images as well as their tags. However, due to the diverse interests of users, the tag distribution is often long-tailed, which can be verified by our analysis in Figure 1 on the tag distribution of the uploaded images at Qzone from January to April in 2016. For the tags in the head part, we can build accurate learners as there are plenty of labeled data but in the tail part, due to the scarce labeled data, the learner for each tag usually has no satisfactory performance. In this case, we can adopt transfer learning algorithms to build accurate classifiers for tags in the tail part by reusing knowledge in the head part. When the tag in the tail part is related to that in the head part, this strategy usually works very well. For example, as shown in Figure 1, we can build an accurate tiger classifier by transferring knowledge from cat images when we have few labeled tiger images, where the performance improvement is as large as 24% compared to some supervised learning algorithm learned from labeled tiger images only. However, if the two tags (e.g., face and airplane images) are totally unrelated from our perspective, existing transfer learning algorithms such as (Patel et al. 2015) fail as shown in Figure 1. One reason for the failure of existing transfer learning algorithms

[1]http://qzone.qq.com

is that the two domains, face and airplane, do not share any common characteristic in shape or other aspects, and hence they are conceptually distant, which violates the assumption of existing transfer learning works that the source domain and the target domain are closely related.

In this paper, we focus on transferring knowledge between two distant domains, which is referred to Distant Domain Transfer Learning (DDTL). The DDTL problem is critical as solving it can largely expand the application scope of transfer learning and help reuse as much previous knowledge as possible. Nonetheless, this is a difficult problem as the distribution gap between the source domain and the target domain is large. The motivation behind our solution to solve the DDTL problem is inspired by human's 'transitivity' learning and inference ability (Bryant and Trabasso 1971). That is, people transfer knowledge between two seemingly unrelated concepts via one or more intermediate concepts as a bridge.

Along this line, there are several works aiming to solve the DDTL problem. For instance, Tan et al. (2015) introduce annotated images to bridge the knowledge transfer between text data in the source domain and image data in the target domain, and Xie et al. (2016) predict the poverty based on the daytime satellite imagery by transferring knowledge from an object classification task with the help of some nighttime light intensity information as an intermediate bridge. Those studies assume that there is only one intermediate domain and that all the data in the intermediate domain are helpful. However, in some cases the distant domains can only be related via multiple intermediate domains. Exploiting only one intermediate domain is not enough to help transfer knowledge across long-distant domains. Moreover, given multiple intermediate domains, it is highly possible that only a subset of data from each intermediate domain is useful for the target domain, and hence we need an automatic selection mechanism to determine the subsets.

In this paper, to solve the DDTL problem in a better way, we aim to transfer knowledge between distant domains by gradually selecting multiple subsets of instances from a mixture of intermediate domains as a bridge. We use the reconstruction error as a measure of distance between two domains. That is, if the data reconstruction error on some data points in the source domain is small based on a model trained on the target domain, then we consider that these data points in the source domain are helpful for the target domain. Based on this measure, we propose a Selective Learning Algorithm (SLA) for the DDTL problem, which simultaneously selects useful instances from the source and intermediate domains, learns high-level representations for selected data, and trains a classifier for the target domain. The learning process of SLA is an iterative procedure that selectively adds new data points from intermediate domains and removes unhelpful data in the source domain to revise the source-specific model changing towards a target-specific model step by step until some stopping criterion is satisfied.

The contributions of this paper are three-fold. Firstly, to our best knowledge, this is the first work that studies the DDTL problem by using a mixture of intermediate domains. Secondly, we propose an SLA algorithm for DDTL. Thirdly, we conduct extensive experiments on several real-world datasets to demonstrate the effectiveness of the proposed algorithm.

Related Work

Typical transfer learning algorithms include instance weighting approaches (Dai et al. 2007) which select relevant data from the source domain to help the learning in the target domain, feature mapping approaches (Pan et al. 2011) which transform the data in both source and target domains into a common feature space where data from the two domains follow similar distributions, and model adaptation approach (Aytar and Zisserman 2011) which adapt the model trained in the source domain to the target domain. However, these approaches cannot handle the DDTL problem as they assume that the source domain and the target domain are conceptually close. Recent studies (Yosinski et al. 2014; Oquab et al. 2014; Long et al. 2015) reveal that deep neural networks can learn transferable features for a target domain from a source domain but they still assume that the target domain is closely related to the source domain.

The transitive transfer learning (TTL) (Tan et al. 2015; Xie et al. 2016) also learns from the target domain with the help of a source domain and an intermediate domain. In TTL, there is only one intermediate domain, which is selected by users manually, and all intermediate domain data are used. Different from TTL, our work automatically selects subsets from a mixture of multiple intermediate domains as a bridge across the source domain and the target domain. Transfer Learning with Multiple Source Domains (TLMS) (Mansour, Mohri, and Rostamizadeh 2009; Tan et al. 2013) leverages multiple source domains to help learning in the target domain, and aims to combine knowledge simultaneously transferred from all the source domains. The different between TLMS and our work is two-fold. First, all the source domains in TLMS have sufficient labeled data. Second, all the source domains in TLMS are close to the target domain.

Self-Taught Learning (STL) (Raina et al. 2007) aims to build a target classifier with limited target-domain labeled data by utilizing a large amount of unlabeled data from different domains to learn a universal feature representation. The difference between STL and our work is two-fold. First, there is no a so-called source domain in STL. Second, STL aims to use all the unlabeled data from different domains to help learning in the target domain while our work aims to identify useful subsets of instances from the intermediate domains to bridge the source domain and the target domain.

Semi-supervised autoencoder (SSA) (Weston, Ratle, and Collobert 2008; Socher et al. 2011) also aims to minimize both the reconstruction error and the training loss while learning a feature representation. However, our work is different from SSA in three-fold. First, in SSA, both unlabeled and labeled data are from the same domain, while in our work, labeled data are from either the source domain or the target domain and unlabeled data are from a mixture of intermediate domains, whose distributions can be very different to each other. Second, SSA uses all the labeled and unlabeled data for learning, while our work selectively chooses some unlabeled data from the intermediate domains and removes some labeled data from the source domain for assisting the

learning in the target domain. Third, SSA does not have convolutional layer(s), while our work uses convolutional filters if the input is a matrix or tensor.

Problem Definition

We denote by $\mathcal{S} = \{(\boldsymbol{x}_S^1, y_S^1), \cdots, (\boldsymbol{x}_S^{n_S}, y_S^{n_S})\}$ the source domain labeled data of size n_S, which is assumed to be sufficient enough to train an accurate classifier for the source domain, and by $\mathcal{T} = \{(\boldsymbol{x}_T^1, y_T^1), \cdots, (\boldsymbol{x}_T^{n_T}, y_T^{n_T})\}$ the target domain labeled data of size n_T, which is assumed to be too insufficient to learn an accurate classifier for the target domain. Moreover, we denote by $\mathcal{I} = \{\boldsymbol{x}_I^1, \cdots, \boldsymbol{x}_I^{n_I}\}$ the mixture of unlabeled data of multiple intermediate domains, where n_I is assumed to be large enough. In this work, a domain corresponds to a concept or class for a specific classification problem, such as face or airplane recognition from images. Without loss of generality, we suppose the classification problems in the source domain and the target domain are both binary. All data points are supposed to lie in the same feature space. Let $p_S(\boldsymbol{x}), p_S(y|\boldsymbol{x}), p_S(\boldsymbol{x}, y)$ be the marginal, conditional and joint distributions of the source domain data, respectively, $p_T(\boldsymbol{x}), p_T(y|\boldsymbol{x}), p_T(\boldsymbol{x}, y)$ be the parallel definitions for the target domain, and $p_I(\boldsymbol{x})$ be the marginal distribution for the intermediate domains. In a DDTL problem, we have

$$p_T(\boldsymbol{x}) \neq p_S(\boldsymbol{x}),\ p_T(\boldsymbol{x}) \neq p_I(\boldsymbol{x}),\ \text{and}\ p_T(y|\boldsymbol{x}) \neq p_S(y|\boldsymbol{x}).$$

The goal of DDTL is to exploit the unlabeled data in the intermediate domains to build a bridge between the source and target domains, which are originally distant to each other, and train an accurate classifier for the target domain by transferring supervised knowledge from the source domain with the help of the bridge. Note that not all the data in the intermediate domains are supposed to be similar to the source domain data, and some of them may be quite different. Therefore, simply using all the intermediate data to build the bridge may fail to work.

The Selective Learning Algorithm

In this section, we present the proposed SLA.

Auto-Encoders and Its Variant

As a basis component in our proposed method to solve the DDTL problem is the autoencoder (Bengio 2009) and its variant, we first review them. An autoencoder is an unsupervised feed-forward neural network with an input layer, one or more hidden layers, and an output layer. It usually includes two processes: encoding and decoding. Given an input $\boldsymbol{x} \in \mathbb{R}^q$, an autoencoder first encodes it through an encoding function $f_e(\cdot)$ to map it to a hidden representation, and then decodes it through a decoding function $f_d(\cdot)$ to reconstruct $\boldsymbol{x}$. The process of the autoencoder can be summarized as

$$\text{encoding}: \boldsymbol{h} = f_e(\boldsymbol{x}),\ \text{and decoding}: \hat{\boldsymbol{x}} = f_d(\boldsymbol{h}),$$

where $\hat{\boldsymbol{x}}$ is the reconstructed input to approximate $\boldsymbol{x}$. The learning of the pair of encoding and decoding functions, $f_e(\cdot)$ and $f_d(\cdot)$, is done by minimizing the reconstruction error over all training data, i.e., $\min_{f_e, f_d} \sum_{i=1}^{n} \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|_2^2$.

After the pair of encoding and decoding functions are learned, the output of encoding function of an input $\boldsymbol{x}$, i.e., $\boldsymbol{h} = f_e(\boldsymbol{x})$, is considered as a higher-level and robust representation for $\boldsymbol{x}$. Note that an autoencoder takes a vector as the input. When an input instance represented by a matrix or tensor, such as images, is presented to an autoencoder, the spatial information of the instance may be discarded. In this case, a convolutional autoencoder is more desired, and it is a variant of the autoencoder by adding one or more convolutional layers to capture inputs, and one or more correspondingly deconvolutional layers to generate outputs.

Instance Selection via Reconstruction Error

A motivation behind our proposed method is that in an ideal case, if the data from the source domain are similar and useful for the target domain, then one should be able to find a pair of encoding and decoding functions such that the reconstruction errors on the source domain data and the target domain data are both small. In practice, as the source domain and the target domain are distant, there may be only a subset of the source domain data is useful for the target domain. The situation is similar in the intermediate domains. Therefore, to select useful instances from the intermediate domains, and remove irrelevant instances from the source domain for the target domain, we propose to learn a pair of encoding and decoding functions by minimizing reconstruction errors on the selected instances in the source and intermediate domains, and all the instances in the target domain simultaneously. The objective function to be minimized is formulated as follows:

$$\begin{aligned}\mathcal{J}_1(f_e, f_d, \boldsymbol{v}_S, \boldsymbol{v}_T) = &\frac{1}{n_S}\sum_{i=1}^{n_S} v_S^i \|\hat{\boldsymbol{x}}_S^i - \boldsymbol{x}_S^i\|_2^2 + \frac{1}{n_I}\sum_{i=1}^{n_I} v_I^i \|\hat{\boldsymbol{x}}_I^i - \boldsymbol{x}_I^i\|_2^2 \\ &+ \frac{1}{n_T}\sum_{i=1}^{n_T} \|\hat{\boldsymbol{x}}_T^i - \boldsymbol{x}_T^i\|_2^2 + R(\boldsymbol{v}_S, \boldsymbol{v}_T), \quad (1)\end{aligned}$$

where $\hat{\boldsymbol{x}}_S^i$, $\hat{\boldsymbol{x}}_I^i$ and $\hat{\boldsymbol{x}}_T^i$ are reconstructions of $\boldsymbol{x}_S^i$, $\boldsymbol{x}_I^i$ and $\boldsymbol{x}_T^i$ based on the autoencoder, $\boldsymbol{v}_S = (v_S^1, \cdots, v_S^{n_S})^\top$, $\boldsymbol{v}_I = (v_I^1, \cdots, v_I^{n_I})^\top$, and $v_S^i, v_I^j \in \{0, 1\}$ are selection indicators for the i-th instance in the source domain and the j-th instance in the intermediate domains, respectively. When the value is 1, the corresponding instance is selected and otherwise unselected. The last term in the objective, $R(\boldsymbol{v}_S, \boldsymbol{v}_T)$, is a regularization term on $\boldsymbol{v}_S$ and $\boldsymbol{v}_T$ to avoid a trivial solution by setting all values of $\boldsymbol{v}_S$ and $\boldsymbol{v}_T$ to be zero. In this paper, we define $R(\boldsymbol{v}_S, \boldsymbol{v}_T)$ as

$$R(\boldsymbol{v}_S, \boldsymbol{v}_T) = -\frac{\lambda_S}{n_S}\sum_{i=1}^{n_S} v_S^i - \frac{\lambda_I}{n_I}\sum_{i=1}^{n_I} v_I^i.$$

Minimizing this term is equivalent to encouraging to select many instances as possible from the source and intermediate domains. Two regularization parameters, λ_S and λ_I, control the importance of this regularization term. Note that more useful instances are selected, more robust hidden representations can be learned through the autoencoder.

Incorporation of Side Information

By solving the minimization problem (1), one can select useful instances from the source and intermediate domains for the target domain through $\boldsymbol{v}_S$ and $\boldsymbol{v}_T$, and learn high-level

hidden representations for data in different domains through the encoding function, i.e., $\boldsymbol{h}=f_e(\boldsymbol{x})$, simultaneously. However, the learning process is in an unsupervised manner. As a result, the learned hidden representations may not be relevant to the classification problem in the target domain. This motivates us to incorporate side information into the learning of the hidden representations for different domains. For the source and target domains, labeled data can be used as the side information, while for the intermediate domains, there is no label information. In this work, we consider the predictions on the intermediate domains as the side information, and use the confidence on the predictions to guide the learning of the hidden representations. To be specific, we propose to incorporate the side information into learning by minimizing the following function:

$$\mathcal{J}_2(f_c,f_e,f_d)=\frac{1}{n_S}\sum_{i=1}^{n_S}v_S^i\ell(y_S^i,f_c(\boldsymbol{h}_S^i))+\frac{1}{n_T}\sum_{i=1}^{n_T}\ell(y_T^i,f_c(\boldsymbol{h}_T^i)) +\frac{1}{n_I}\sum_{i=1}^{n_I}v_I^i g(f_e(\boldsymbol{h}_I^i)), \tag{2}$$

where $f_c(\cdot)$ is a classification function to output classification probabilities, and $g(\cdot)$ is entropy function defined as $g(z)=-z\ln z-(1-z)\ln(1-z)$ for $0\le z\le 1$, which is used to select instances of high prediction confidence in the intermediate domains.

Overall Objective Function

By combining the two objectives in Eqs. (1) and (2), we obtain the final objective function for DDTL as follows:

$$\min_{\boldsymbol{\Theta},\boldsymbol{v}}\mathcal{J}=\mathcal{J}_1+\mathcal{J}_2,\qquad \text{s.t. } v_S^i,v_I^i\in\{0,1\}, \tag{3}$$

where $\boldsymbol{v}=\{\boldsymbol{v}_S,\boldsymbol{v}_T\}$, and $\boldsymbol{\Theta}$ denotes all parameters of the functions $f_c(\cdot)$, $f_e(\cdot)$, and $f_d(\cdot)$.

To solve problem (3), we use the block coordinate decedent (BCD) method, where in each iteration, variables in each block are optimized sequentially while keeping other variables fixed. In problem (3), there are two blocks of variables: $\boldsymbol{\Theta}$ and $\boldsymbol{v}$. When the variables in $\boldsymbol{v}$ are fixed, we can update $\boldsymbol{\Theta}$ using the Back Propagation (BP) algorithm where the gradients can be computed easily. Alternatingly, when the variables in $\boldsymbol{\Theta}$ are fixed, we can obtain an analytical solution for $\boldsymbol{v}$ as follows,

$$v_S^i=\begin{cases}1 & \text{if } \ell(y_s^i,f_c(f_e(\boldsymbol{x}_S^i)))+\|\hat{\boldsymbol{x}}_S^i-\boldsymbol{x}_S^i\|_2^2<\lambda_S\\ 0 & \text{otherwise}\end{cases} \tag{4}$$

$$v_I^i=\begin{cases}1 & \text{if } \|\hat{\boldsymbol{x}}_I^i-\boldsymbol{x}_I^i\|_2^2+g(f_c(f_e(\boldsymbol{x}_I^i)))<\lambda_I\\ 0 & \text{otherwise}\end{cases} \tag{5}$$

Based on Eq. (4), we can see that for the data in the source domain, only those with low reconstruction errors and low training losses will be selected during the optimization procedure. Similarly, based on Eq. (5), it can be found that for the data in the intermediate domains, only those with low reconstruction errors and high prediction confidence will be selected.

An intuitive explanation of this learning strategy is twofold: 1) When updating $\boldsymbol{v}$ with a fixed $\boldsymbol{\Theta}$, "useless" data in the source domain will be removed, and those intermediate data that can bridge the source and target domains will be selected for training; 2) When updating $\boldsymbol{\Theta}$ with fixed $\boldsymbol{v}$, the model is trained only on the selected "useful" data samples. The overall algorithm for solving problem (3) is summarized in Algorithm 1.

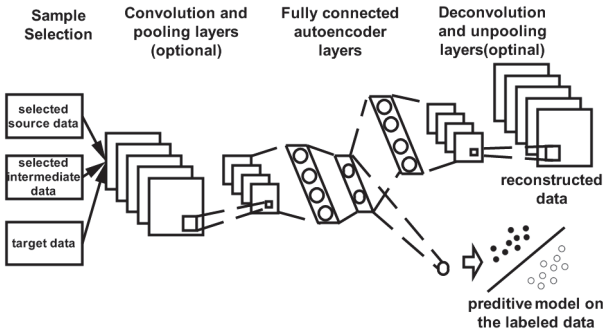


Figure 2: The deep architecture for SAN/SCAN models.

The deep learning architecture corresponding to problem (3) is illustrated in Figure 2. From Figure 2, we note that except for the instance selection component $\boldsymbol{v}$, the rest of the architecture in Figure 2 can be viewed as a generalization of an autoencoder or a convolutional autoencoder by incorporating the side information, respectively. In the sequel, we refer to the architecture (except for the instances selection component) using an antoencoder for $f_e(\cdot)$ and $f_d(\cdot)$ as SAN (Supervised AutoeNcoder), and the one using a convolutional antoencoder as SCAN (Supervised Convolutional AutoeNcoder).

Algorithm 1 The Selective Learning Algorithm (SLA)

1: **Input**: Data in $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{I}$, and parameters λ_S, λ_I, and T;
2: Initialize $\boldsymbol{\Theta}$, $\boldsymbol{v}_S=\mathbf{1}$, $\boldsymbol{v}_I=\mathbf{0}$; *// All source data are used*
3: **while** $t<T$ **do**
4: Update $\boldsymbol{\Theta}$ via the BP algorithm; *// Update the network*
5: Update $\boldsymbol{v}$ by Eqs. (4) and (5); *// Select "useful" instances*
6: $t=t+1$
7: **end while**
8: **Output**: $\boldsymbol{\Theta}$ and $\boldsymbol{v}$.

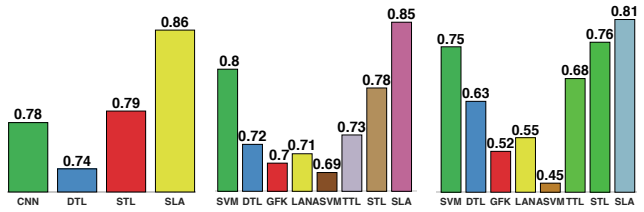
Experiments

In this section, we conduct empirical studies to evaluate the proposed SLA algorithm from three aspects.[2] Firstly, we test the effectiveness of the algorithm when the source domain and the target domain are distant. Secondly, we visualize some selected intermediate data to understand how the algorithm works. Thirdly, we evaluate the importance of the learning order of intermediate instances by comparing the order generated by SLA with other manually designed orders.

Baseline Methods

Three categories of methods are used as baselines.

[2]We will use SLA to denote SLA-SAN or SLA-SCAN when there is no confusion.



(a) Caltech (Tensor) (b) Caltech (SIFT) (c) AwA (SIFT)

Figure 3: Average accuracies of different learning algorithms on the Caltech-256 and AwA datasets.

Supervised Learning In this category, we choose two supervised learning algorithms, SVM and convolutional neural networks (CNN) (Krizhevsky, Sutskever, and Hinton 2012), as baselines. For SVM, we use the linear kernel. For CNN, we implement a network that is composed of two convolutional layers with kernel size 3×3, where each convolutional layer is followed by a max pooling layer with kernel size 2×2, a fully connected layer, and a logistic regression layer.

Transfer Learning In this category, we choose five transfer learning algorithms including Adaptive SVM (ASVM) (Aytar and Zisserman 2011), Geodesic Flow Kernel (GFK) (Gong et al. 2012), Landmark (LAN) (Gong, Grauman, and Sha 2013), Deep Transfer Learning (DTL) (Yosinski et al. 2014), and Transitive Transfer Learning (TTL) (Tan et al. 2015). For ASVM, we use all the source domain and target domain labeled data to train a target model. For GFK, we first generate a series of intermediate spaces using all the source domain and target domain data to learn a new feature space, and then train a model in the space using all the source domain and target domain labeled data. For LAN, we first select a subset of labeled data from the source domain, which has a similar distribution as the target domain, and then use all the selected data to facilitate knowledge transfer. For DTL, we first train a deep model in the source domain, and then train another deep model for the target domain by reusing the first several layers of the source model. For TTL, we use all the source domain, target domain and intermediate domains data to learn a model.

Self-taught Learning (STL) We apply the autoencoder or convolutional autoencoder on all the intermediate domains data to learn a universal feature representation, and then train a classifier with all the source domain and target domain labeled data with the new feature representation.

Among the baselines, CNN can receive tensors as inputs, DTL and STL can receive both vectors and tensors as inputs, while the other models can only receive vectors as inputs. For a fair comparison, in experiments, we compare the proposed SLA-SAN method with SVM, GFK, LAN, ASVM, DTL, STL and TTL, while compare the SLA-SCAN method with CNN, DTL and STL. The convolutional autoencoder component used in SCAN has the same network structure as the CNN model, except that the fully connected layer is connected to two unpooling layers and two deconvolutional layers to reconstruct inputs. For all deep-learning based models, we use all the training data for pre-training.

Table 1: Accuracies (%) of selected tasks on Catech-256.

	CNN	DTL	STL	SLA-SCAN
'face-to-horse'	72 ± 2	67 ± 2	70 ± 3	$\mathbf{89} \pm 2$
'horse-to-airplane'	82 ± 2	76 ± 3	82 ± 2	$\mathbf{92} \pm 2$
'gorilla-to-face'	83 ± 1	80 ± 2	82 ± 2	$\mathbf{96} \pm 1$

Datasets

The datasets used for experiments include Caltech-256 (Griffin, Holub, and Perona 2007) and Animals with Attributes (AwA)[3]. The Caltech-256 dataset is a collection of 30,607 images from 256 object categories, where the number of instances per class varies from 80 to 827. Since we need to transfer knowledge between distant categories, we choose a subset of categories including '*face*', '*watch*', '*airplane*', '*horse*', '*gorilla*', '*billiards*' to form the source and target domains. Specifically, we first randomly choose one category to form the source domain, and consider images belonging to this category as positive examples for the source domain. Then we randomly choose another category to form the target domain, and consider images in this category as positive examples for the target domain. As this dataset has a *clutter* category, we sample images from this category as negative samples for the source and target domains and the sampling process guarantees that each pair of source and target domains has no overlapping on negative samples. After constructing the source domain and the target domain, all the other images in the dataset are used as the intermediate domains data. Therefore, we construct $P_6^2 = 30$ pairs of DDTL problems from this dataset.

The AwA dataset consists of 30,475 images of 50 animals classes, where the number of samples per class varies from 92 to 1,168. Due to the copyright reason, this dataset only provides extracted SIFT features for each image instead of the original images. We choose three categories including 'humpback+whale', 'zebra' and 'collie' to form the source and target domains. The source-target domains construction procedure is similar to that on the Caltech-256 dataset, and images in the categories 'beaver', 'blue+whale', 'mole', 'mouse', 'ox', 'skunk', and 'weasel' are considered as negative samples for the source domain and the target domain. In total we construct $P_3^2 = 6$ pairs of DDTL problems.

Performance Comparison

In experiments, for each target domain, we randomly sample 6 labeled instances for training, and use the rest for testing. Each configuration is repeated 10 times. On the Caltech-256 dataset, we conduct two sets of experiments. The first experiment uses the original images as inputs to compare SLA-SCAN with CNN, DTL and STL. The second experiment uses SIFT features extracted from images as inputs to do a comparison among SLA-SAN and the rest baselines. On the AwA dataset, since only SIFT features are given, we compare SLA-SAN with SVM, GFK, LAN, ASVM, DTL, STL and TTL. The comparison results in terms of average accuracy are shown in Figure 3. From the results, we can see

[3] http://attributes.kyb.tuebingen.mpg.de/

Table 2: Accuracies (%) of selected tasks on Catech-256 and AwA with SIFT features.

	SVM	DTL	GFK	LAN	ASVM	TTL	STL	SLA
'horse-to-face'	84 ± 2	88 ± 2	77 ± 3	79 ± 2	76 ± 4	78 ± 2	86 ± 3	$\mathbf{92} \pm 2$
'airplane-to-gorilla'	75 ± 1	62 ± 3	67 ± 5	66 ± 4	51 ± 2	65 ± 2	76 ± 3	$\mathbf{84} \pm 2$
'face-to-watch'	75 ± 7	68 ± 3	61 ± 4	63 ± 4	60 ± 5	67 ± 4	75 ± 5	$\mathbf{88} \pm 4$
'zebra-to-collie'	71 ± 3	69 ± 2	56 ± 2	57 ± 3	59 ± 2	70 ± 3	72 ± 3	$\mathbf{76} \pm 2$

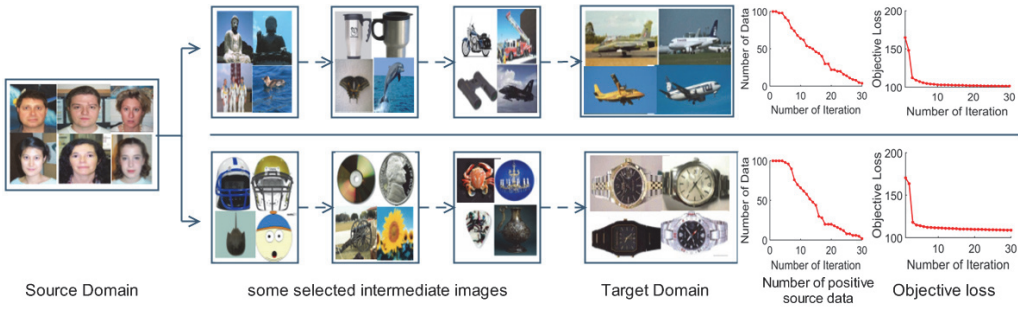
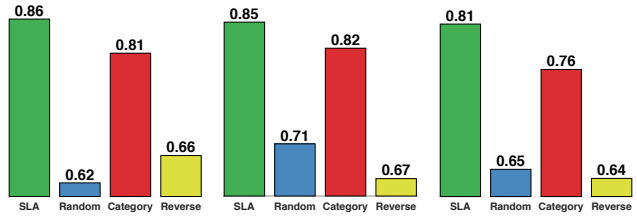


Figure 4: Detailed results: visualization of the selected intermediate data over iterations, the change of the remaining source data size over iterations, and the change of the objective function value over iterations in two transfer learning tasks: 'face-to-airplane' and 'face-to-watch'.

that the transfer learning methods such as DTL, GFK, LAN and ASVM achieve worse performance than CNN or SVM because the source domain and the target domain have huge distribution gap, which leads to 'negative transfer'. TTL does not work either due to the distribution gap among the source, intermediate and target domains. STL achieves slightly better performance than the supervised learning methods. A reason is that the universal feature representation learned from the intermediate domain data helps the learning of the target model to some extent. Our proposed SLA method obtains the best performance under all the settings. We also report the average accuracy as well as standard deviations of different methods on some selected tasks in Tables 1 and 2. On these tasks, we can see that for SLA-SCAN, the improvement in accuracy over CNN is larger than 10% and sometimes up to 17%. For SLA-SAN, the improvement in accuracy is around 10% over SVM.

Comparison on Different Learning Orders

As shown in Figure 4, the SLA algorithm can learn an order of useful intermediate data to be added into the learning process. To compare different orders of intermediate data to be added into learning, we manually design three ordering strategies. In the first strategy, denoted by Random, we randomly split intermediate domains data into ten subsets. In each iteration, we add one subset into the learning process. In the second strategy, denoted by Category, we first obtain an order using the SLA algorithm, and then use the additional category information of the intermediate data to add the intermediate data category by category into the learning process. Hence the order of the categories is the order they first appear in the order learned by the SLA algorithm. The third strategy, denoted by Reverse, is similar to the second one except that the order of the categories to be added is reversed. In the



(a) Caltech (Tensor) (b) Caltech (SIFT) (c) AwA (SIFT)

Figure 5: Comparison results with different learning orders on the Caltech-256 and AwA datasets.

experiments, the source domain data are removed in the same way as the original learning process of the SLA algorithm. From the results shown in Figure 5, we can see that the three manually designed strategies obtain much worse accuracy than SLA. Furthermore, among the three strategies, 'Category' achieves the best performance, which is because this order generation is close to that of SLA.

Detailed Results

To understand how the data in intermediate domains can help connect the source domain and the target domain, in Figure 4, we show some images from intermediate domains selected in different iterations of the SLA algorithm on two transfer learning tasks, 'face-to-airplane' and 'face-to-watch', on the Caltech-256 dataset. Given the same source domain 'face' and two different target domains 'airplane' and 'watch', in Figure 4, we can see that the model selects completely different data points from intermediate domains. It may be not easy to figure out why those images are iteratively selected from our perspective. However, intuitively we can find that at the beginning, the selected images are closer to the source

images such as 'buddhas' and 'football-helmet' in the two tasks, respectively, and at the end of the learning process, the selected images look closer to the target images. Moreover, in Figure 4, we show that as the iterative learning process proceeds, the number of positive samples in the source domain involved decreases and the value of the objective function in problem (3) decreases as well.

Conclusion

In this paper, we study the novel DDTL problem, where the source domain and the target domain are distant, but can be connected via some intermediate domains. To solve the DDTL problem, we propose the SLA algorithm to gradually select unlabeled data from the intermediate domains to bridge the two distant domains. Experiments conducted on two benchmark image datasets demonstrate that SLA is able to achieve state-of-the-art performance in terms of accuracy. As a future direction, we will extend the proposed algorithm to handle multiple source domains.

Acknowledgments

We are supported by National Grant Fundamental Research (973 Program) of China under Project 2014CB340304, Hong Kong CERG projects 16211214, 16209715 and 16244616, and Natural Science Foundation of China under Project 61305071. Sinno Jialin Pan thanks the support from the NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020. We thank Hao Wang for helpful discussions and reviewers for their valuable comments.

References

Aytar, Y., and Zisserman, A. 2011. Tabula Rasa: Model transfer for object category detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2252–2259.

Bengio, Y. 2009. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning* 2(1):1–127.

Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Wortman, J. 2008. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems 21*, 129–136.

Bryant, P. E., and Trabasso, T. 1971. Transitive inferences and memory in young children. *Nature*.

Dai, W.; Yang, Q.; Xue, G.-R.; and Yu, Y. 2007. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, 193–200.

Gong, B.; Shi, Y.; Sha, F.; and Grauman, K. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2066–2073.

Gong, B.; Grauman, K.; and Sha, F. 2013. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *Proceedings of the 30th International Conference on Machine Learning*, 222–230.

Griffin, G.; Holub, A.; and Perona, P. 2007. Caltech-256 object category dataset. Technical Report CNS-TR-2007-001, California Institute of Technology.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 1097–1105.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. I. 2015. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on Machine Learning*, 97–105.

Mansour, Y.; Mohri, M.; and Rostamizadeh, A. 2009. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems 22*, 1041–1048.

Oquab, M.; Bottou, L.; Laptev, I.; and Sivic, J. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1717–1724.

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10).

Pan, S. J.; Tsang, I. W.; Kwok, J. T.; and Yang, Q. 2011. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks* 22(2):199–210.

Patel, V. M.; Gopalan, R.; Li, R.; and Chellappa, R. 2015. Visual domain adaptation: A survey of recent advances. *IEEE Signal Processing Magazine* 32(3):53–69.

Raina, R.; Battle, A.; Lee, H.; Packer, B.; and Ng, A. Y. 2007. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*, 759–766.

Rosenstein, M. T.; Marx, Z.; Kaelbling, L. P.; and Dietterich, T. G. 2005. To transfer or not to transfer. In *NIPS Workshop on Inductive Transfer: 10 Years Later*.

Socher, R.; Pennington, J.; Huang, E. H.; Ng, A. Y.; and Manning, C. D. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 151–161.

Tan, B.; Zhong, E.; Xiang, E.; and Yang, Q. 2013. Multi-transfer: Transfer learning with multiple views and multiple sources. In *the 13rd SIAM International Conference on Data Mining*.

Tan, B.; Song, Y.; Zhong, E.; and Yang, Q. 2015. Transitive transfer learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1155–1164.

Weston, J.; Ratle, F.; and Collobert, R. 2008. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning*.

Xie, M.; Jean, N.; Burke, M.; Lobell, D.; and Ermon, S. 2016. Transfer learning from deep features for remote sensing and poverty mapping. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*, 3320–3328.