Artificial Intelligence 275 (2019) 310–328

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

A deep learning framework for Hybrid Heterogeneous Transfer Learning

Joey Tianyi Zhou [a], Sinno Jialin Pan [b,*], Ivor W. Tsang [c]

[a] Institute of High Performance Computing, Singapore
[b] Nanyang Technological University, Singapore
[c] CAI, University of Technology Sydney, Australia

ARTICLE INFO

Article history:
Received 8 December 2018
Received in revised form 25 March 2019
Accepted 4 June 2019
Available online 6 June 2019

Keywords:
Heterogeneous transfer learning
Deep learning
Multilingual text classification

ABSTRACT

Most previous methods in heterogeneous transfer learning learn a cross-domain feature mapping between different domains based on some cross-domain instance-correspondences. Such instance-correspondences are assumed to be representative in the source domain and the target domain, respectively. However, in many real-world scenarios, this assumption may not hold. As a result, the constructed feature mapping may not be precise, and thus the transformed source-domain labeled data using the feature mapping are not useful to build an accurate classifier for the target domain. In this paper, we offer a new heterogeneous transfer learning framework named Hybrid Heterogeneous Transfer Learning (HHTL), which allows the selection of corresponding instances across domains to be biased to the source or target domain. Our basic idea is that though the corresponding instances are biased in the original feature space, there may exist other feature spaces, projected onto which, the corresponding instances may become unbiased or representative to the source domain and the target domain, respectively. With such a representation, a more precise feature mapping across heterogeneous feature spaces can be learned for knowledge transfer. We design several deep-learning-based architectures and algorithms that enable learning aligned representations. Extensive experiments on two multilingual classification datasets verify the effectiveness of our proposed HHTL framework and algorithms compared with some state-of-the-art methods.

1. Introduction

Transfer learning or domain adaptation is an important and promising machine learning paradigm, which aims to transfer knowledge extracted from an auxiliary domain, i.e., the source domain, where sufficient labeled data are available, to solve learning problems in a new domain, i.e., the target domain, with little or no additional human supervision [1]. Recently, more and more attention has been shifted from transferring knowledge across homogeneous domains to transferring knowledge across heterogeneous domains, where the source domain and the target domain have heterogeneous types of features [2,3]. In contrast with homogeneous transfer learning, which assumes that the source domain data and the target domain data are represented in the same feature space of the same dimensionality [4,5], and thus the domain difference is only caused by bias in features or data distributions, heterogeneous transfer learning allows the source domain data and the

* Corresponding author.
E-mail addresses: joey.tianyi.zhou@gmail.com (J.T. Zhou), sinnopan@ntu.edu.sg (S.J. Pan), ivor.tsang@gmail.com (I.W. Tsang).

https://doi.org/10.1016/j.artint.2019.06.001
0004-3702/

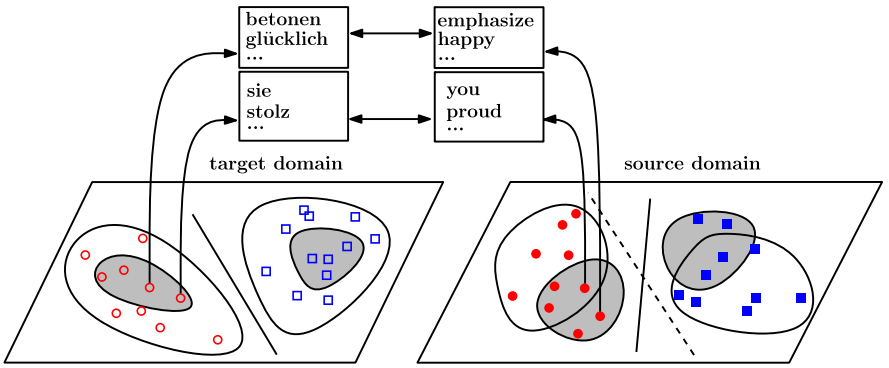


Fig. 1. HHTL Case I: Cross-language bias by translation. Consider a binary classification problem, where circles are of one class, and squares are of the other class. The points with gray background in the source domain (English) are the selected representatives, while those with gray background in the target domain (German) are their correspondences via translation or a feature map learned from English and German corresponding documents. In this case, representatives of the source domain are supposed to be randomly selected, but their correspondences in the target domain are biased to the overall population of the target domain.

target domain data to be represented by non-overlapping feature spaces. Heterogeneous transfer learning has been shown to be crucial in many real-world applications. For instance, many Natural Language Processing (NLP) tasks, such as named entity recognition, coreference resolution, etc., highly rely on sufficient annotated corpora and linguistic/semantic knowledge bases to build an accurate classifier. For English, annotated corpora and knowledge bases are widely available, while for other languages, such as Thai, Vietnamese, etc., few resources are available. In this case, heterogeneous transfer learning is desirable to transfer knowledge extracted from the rich English resources to solve NLP tasks in some other languages of poor resources.

Most existing approaches to heterogeneous transfer learning aim to learn a feature mapping across heterogeneous feature spaces based on some cross-domain correspondences constructed either by labels in both the source domain and the target domain [6] or *a translator* between domains [7]. With the learned feature mapping, instances can be mapped from the target domain to the source domain or the other way round. In this way, if the feature mapping is learned precisely, then source-domain labeled data can still be used to learn an accurate classifier for the target domain. A common assumption behind these methods is that the selected instance-correspondences are *representative* to the source domain and the target domain such that a "perfect" cross-domain feature mapping can be learned. However, in many real-world scenarios, this assumption may not hold, which means that the selected corresponding instances may be biased to get the overall population(s) of the source domain or/and the target domain, and thus are not able to represent either the source domain data or the target domain data. As a result, it may fail to learn a precise cross-domain feature mapping based on a relatively small set of cross-domain instance-correspondences.

Consider cross-language document classification as a motivating example as illustrated in Fig. 1. The objective is to learn a text classifier whose goal is to automatically categorize documents in German (i.e., the target domain) only with a set of annotated English documents (i.e., the source domain). To apply heterogeneous transfer learning methods to solve this task, one can first randomly select some German documents, which can be considered as representatives in the German (or source) domain, and then simply construct German-English document-correspondences by translating the selected German documents into English using Google translator. However, the *wordbook* of the translated English documents may be quite different from that of the native English documents. For instance, the German word "betonen" is translated into the English word "emphasize" by Google translator. However, in an English document written by a native English speaker, its corresponding word might be "highlight" or "stress" instead. This is referred to as the "feature bias" issue in word usages between the translated documents and the original ones. Therefore, the translated English documents may be biased or not representative in the English (i.e., source) domain. In this case, a feature mapping learned based on such biased cross-domain correspondences may not be effective for knowledge transfer.

As another example shown in Fig. 2, consider a multilingual sentiment classification task, which is to automatically predict the overall sentiment polarities of song reviews in German (i.e., the target domain), given labeled book reviews in English (i.e., the source domain) as well as some unlabeled book reviews in German. To construct correspondences between the source domain and the target domain, one can randomly select some unlabeled book reviews in German, and translate them into English using Google translator. Even though the selected book reviews in German can be considered as representatives to the German book domain, they are not representative to the German song domain, which is the target domain in this example. This is because opinion and topic words used for different types of products can be very different [8–10]. This type of "feature bias" is caused by the difference of word usage for different types of products. In addition, similarly to the former example, their English translations are not representative in the English book domain either, which is the source domain in this example. This means that the selected instances from the source domain and their corresponding instances in the target domain are biased or not representative to the source domain and the target

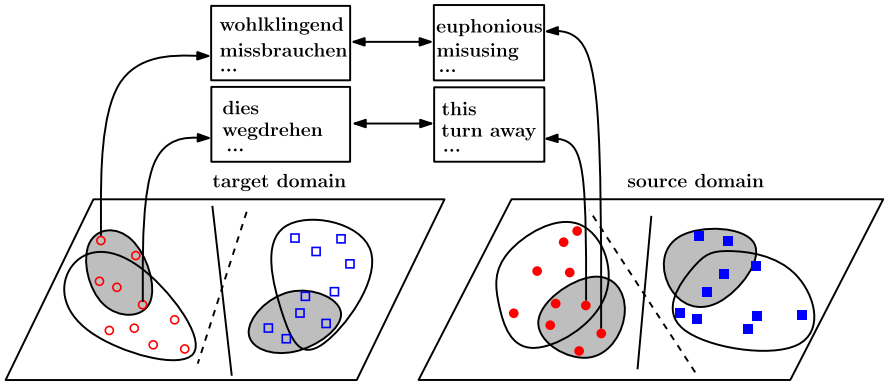


Fig. 2. HHTL Case II: Cross-product and cross-language words bias. Consider a binary classification problem, where circles are of one class, and squares are of the other class. The points with gray background in the source domain (English) are the selected representatives, while those with gray background in the target domain (German) are their correspondences via translation or a feature map learned from English and German corresponding documents. In this case, the selected source-domain and target-domain instances are biased to populations of the source and target domain, respectively.

domain, respectively. As a result, the cross-domain feature mapping learned with such correspondences may not be effective for heterogeneous transfer learning.

Motivated by the above two examples, we propose a new heterogeneous transfer learning framework named "hybrid heterogeneous transfer learning" (HHTL) to transfer knowledge across heterogeneous domains effectively. Specifically, the proposed HHTL framework consists of two main components: 1) learning a heterogeneous feature mapping between the source domain and the target domain based on the pre-constructed instance-correspondences, and 2) discovering a more powerful representation to reduce the feature bias caused by either the difference in homogeneous feature space (e.g., book reviews v.s. song reviews in a same language) or an imprecise *translator* (e.g., English reviews v.s. English translations of German reviews). Our basic idea is that though the corresponding instances are biased in the original feature space, there may exist other feature spaces, projected onto which, respectively, the corresponding instances may become unbiased or representative to the source domain and the target domain. These two components are learned in an iterative manner. Following that, standard classification methods can be applied on the source-domain labeled data with the new representation to build an accurate target classifier. We propose two deep-learning-based architectures to implement the HHTL framework, and develop five solutions to simultaneously learn the feature mapping and high-level features across heterogeneous domains in particular.

Note that the proposed HHTL framework is different from multi-view learning [11–13], where full correspondences between two views of data are required, and the labels of all the correspondences are assumed to be available in general or some of them are assumed to be available in the semi-supervised learning manner [14]. In HHTL, no label information of the cross-domain instance-correspondences is required. Moreover, in HHTL, labeled data are only assumed to be available in the source domain, and no labeled data is required in the target domain, while the goal is to learn a classifier for the target domain.

Compared to our preliminary work [15], the contributions of this paper are summarized as follows.

- We generalize the deep learning solution proposed in [15] to a unified HHTL framework.
- Based on the framework, we design two deep learning architectures, where our preliminary work falls into one of the two proposed architectures. Based on the new architecture proposed in this work, we further propose two specific solutions. In particular, the first solution enjoys closed forms for linear transformations on in-domain and cross-domain feature mappings. The second solution replaces the linear cross-domain mapping in the model by neural networks, which is more powerful for feature learning. The third solution further replaces the in-domain mapping by a neural network and proposes a co-learning objective for in-domain and cross-domain mappings such that all the components can be jointly optimized.
- We conduct more extensive experiments to verify the effectiveness of the proposed HHTL framework and solutions.

2. Deep architectures for Hybrid Heterogeneous Transfer Learning

2.1. Problem formulation

Given a set of target-domain unlabeled data $\mathbf{D}_T = \{\mathbf{x}_{T_i}\}_{i=1}^{n_1}$, a set of source-domain labeled data $\mathbf{D}_S = \{(\mathbf{x}_{S_i}, y_{S_i})\}_{i=1}^{n_2}$, and an additional set of pairs of the source- and target- domain unlabeled data, $\mathbf{D}_C = \{(\mathbf{x}_{S_i}^{(c)}, \mathbf{x}_{T_i}^{(c)})\}_{i=1}^{n_c}$, namely correspondences, where $\mathbf{x}_{S_i}$ and $\mathbf{x}_{S_i}^{(c)}$ are in $\mathbb{R}^{d_S \times 1}$, and $\mathbf{x}_{T_i}$ and $\mathbf{x}_{T_i}^{(c)}$ are in $\mathbb{R}^{d_T \times 1}$. In HTL, the goal is to learn a feature mapping $\mathbf{G}_T$ to map data from the target domain feature space to the source domain feature space, and train the target-domain classifier f from the source-domain labeled data $\mathbf{D}_S$. In this way, making a prediction on any target-domain test data $\mathbf{x}_T^*$ can be done by

performing $f(\mathbf{G}_T\mathbf{x}_T^*)$. Alternatively, one can learn another feature mapping $\mathbf{G}_S$ to map data from the source domain to the target domain, and train the target-domain classifier f' from the mapped source-domain labeled data $\{\mathbf{G}_S\mathbf{x}_{S_i}, y_{S_i}\}_{i=1}^{n_2}$. In this way, making a prediction on any target-domain test data $\mathbf{x}_T^*$ can be done by performing $f'(\mathbf{x}_T^*)$.

For simplicity in presentation, we absorb a constant feature into the feature vector as $\mathbf{x}_S = [\mathbf{x}_S^\top\ 1]^\top$ or $\mathbf{x}_T = [\mathbf{x}_T^\top\ 1]^\top$, and incorporate a bias term $\mathbf{b}_S$ or $\mathbf{b}_T$ within the weight matrix as $\mathbf{W}_S = [\mathbf{W}_S\ \mathbf{b}_S]$ or $\mathbf{W}_T = [\mathbf{W}_T\ \mathbf{b}_T]$. We further denote by $\overline{\mathbf{X}}_S = \left[\mathbf{X}_S\ \mathbf{X}_S^{(c)}\right]$ the union source-domain feature vectors of the source-domain labeled and corresponding unlabeled data,[1] and $\overline{\mathbf{X}}_T = \left[\mathbf{X}_T\ \mathbf{X}_T^{(c)}\right]$ the union target-domain feature vectors of the target-domain unlabeled data without correspondences and those with correspondences.

Formally speaking, as the cross-domain correspondences $\mathbf{D}_C$ are not representative to the source domain data $\mathbf{D}_S$ and/or the target domain data $\mathbf{D}_T$, the feature mapping $\mathbf{G}_T$ (or $\mathbf{G}_S$) learned from $\mathbf{D}_C$ may not be effective for knowledge transfer across heterogeneous domains. Therefore, we aim to learn the feature mapping to minimize the difference between the mapped target (or source) domain data $\mathbf{G}_T\mathbf{X}_T$ (or $\mathbf{G}_S\mathbf{X}_S$) and the source (or target) domain data $\mathbf{X}_S$ (or $\mathbf{X}_T$). In this paper, we term this type of learning problem as **Hybrid Heterogeneous Transfer Learning (HHTL)**.

2.2. Stacked Denoised Autoencoder and its extension

As our proposed deep learning approaches are built on top of Stacked Denoised Autoencoder (SDA), in this section, we briefly review this method and one of its extensions, Marginalized SDA (mSDA). SDA firstly randomly sets some values of the source domain features to be zero, which is referred to as a "corruption" of the source domain data. In total, one can obtain m different corruptions. After that SDA tries to learn high-level features by reconstructing these m corruptions. For example, German word "betonen" is translated to "emphasize" by using Google Translator. However in human writings, one may use the English words "highlight" and "stress" on the context instead of "emphasize" to express the meaning of the German word "betonen". SDA aims to reconstruct the machine translated word "emphasize" by using the words "highlight" and "stress". Therefore, the learned high-level features have capability to reduce feature bias.

In particular, we adopt mSDA for high-level feature learning for instances of homogeneous features. mSDA is an extension of SDA, which simplifies the reconstruction from two-level encoder and decoder to a single mapping. The reasons why we use mSDA are two folds: 1) the effectiveness of mSDA has been shown in homogeneous domain adaptation problems [16], and 2) compared to the standard SDA method, mSDA has proven to be much more efficient.

We denote by $\mathbf{X} \in \mathbb{R}^{d\times n}$ the original raw data of n instances and d features. For simplicity in presentation, following the notations used in [16], we absorb a constant feature into the feature vector as $\mathbf{x} = [\mathbf{x}^\top\ 1]^\top$, and incorporate a bias term $\mathbf{b}$ within the weight matrix as $\mathbf{W} = [\mathbf{W}\ \mathbf{b}]$. The objective of mSDA is to learn a weight matrix $\mathbf{W} \in \mathbb{R}^{(d+1)\times(d+1)}$ by minimizing the squared reconstruction loss as follows,

$$\sum_{i=1}^{m} \left\| \mathbf{X} - \mathbf{W}\mathbf{X}^{(i)} \right\|_F^2, \tag{1}$$

where $\mathbf{X}^{(i)}$ denotes the i-th corrupted version of $\mathbf{X}$. The solution to (1) depends on how the original features are corrupted. Denote by $\widehat{\mathbf{X}}_S = [\mathbf{X}\,\mathbf{X}\,\cdots\,\mathbf{X}]$ the m-times repeated version of $\mathbf{X}$, and $\widetilde{\mathbf{X}}_S$ the corrupted version of $\widehat{\mathbf{X}}_S$. The objective (1) can be written as

$$\mathrm{tr}\left[\left(\widehat{\mathbf{X}} - \mathbf{W}\widetilde{\mathbf{X}}\right)^\top \left(\widehat{\mathbf{X}} - \mathbf{W}\widetilde{\mathbf{X}}\right) \right]. \tag{2}$$

The solution to minimization of (2) can be explicitly expressed as follows,

$$\mathbf{W} = \mathbf{P}\mathbf{Q}^{-1} \quad \text{with} \quad \mathbf{Q} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^\top \quad \text{and} \quad \mathbf{P} = \widehat{\mathbf{X}}\widetilde{\mathbf{X}}^\top.$$

In general, to alleviate bias in estimation, a large number of corrupted replicates of the training data is required, which is computationally expensive. To address this issue, mSDA introduces a corruption probability p to model infinite corruptions, i.e., $m \longrightarrow \infty$. Define a feature vector $\mathbf{q} = [1-p, \cdots, 1-p, 1]^\top \in \mathbb{R}^{d+1}$, where $\mathbf{q}_i$ represents the probability of a feature indexed by i "surviving" after the corruption. Thus, we can obtain the expectation of (2), and its solution can be written analytically as

$$\mathbf{W} = \mathbb{E}[\mathbf{P}]\mathbb{E}[\mathbf{Q}]^{-1}, \tag{3}$$

where $\mathbb{E}[\mathbf{P}]_{ij} = \mathbf{S}_{ij}\mathbf{q}_j$, $\mathbf{S} = \mathbf{X}\mathbf{X}^\top$, and

[1] Note that in practice, if there is an additional set of unlabeled data in the source domain without correspondences in the target domain, one can use it as well for higher-level feature learning. However, in this paper, for simplicity in description, we do not assume that additional source-domain unlabeled data is available.

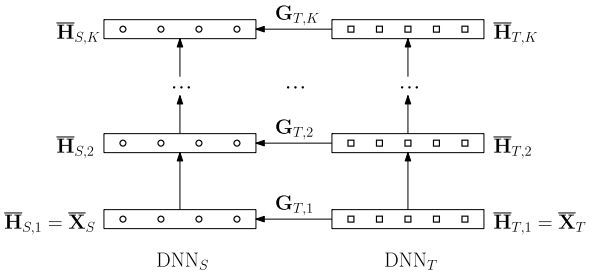


Fig. 3. HHTL: Deep Architecture I.

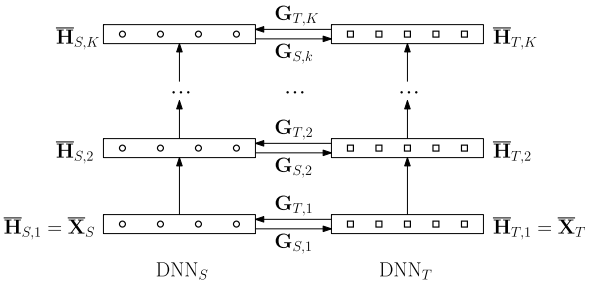


Fig. 4. HHTL: Deep Architecture II.

$$\mathbb{E}[\mathbf{Q}]_{ij} = \begin{cases} \mathbf{S}_{ij}\mathbf{q}_i\mathbf{q}_j, & \text{if } i \neq j, \\ \mathbf{S}_{ij}\mathbf{q}_i, & \text{otherwise.} \end{cases} \tag{4}$$

After $\mathbf{W}$ is learned, one can apply a nonlinear squashing-function, e.g., the hyperbolic tangent function $\tanh(\cdot)$ used in this paper, on the outputs of mSDA to generate nonlinear features as follows,

$$\mathbf{H} = \tanh(\mathbf{W}\mathbf{X}). \tag{5}$$

3. Proposed deep learning approaches for HHTL

In this section, we present the motivation and two overall architectures of our proposed deep learning framework for HHTL. Our motivation is that an intrinsic reason behind the bias issue of the instance-correspondences may be caused by the originally low-level feature representation, e.g., specific words used for expression. To address it, we seek a high-level semantic feature representation for the source (or target) domain, with which the bias issue can be addressed. To facilitate the knowledge transfer, in each layer k, one can then learn a cross-domain feature mapping $\mathbf{G}_{T,k}$ or a pair of cross-domain feature mappings $\mathbf{G}_{T,k}$ and $\mathbf{G}_{S,k}$ with the cross-domain instance-correspondences with high-level feature representations, as shown in Fig. 3 and Fig. 4.

3.1. Approaches based on Deep Architecture I

In this section, we propose a family of two approaches based on Deep Architecture I shown in Fig. 3. The first approach is to construct each $\mathbf{G}_{T,k}$ by learning a linear feature mapping from the high-level feature $\mathbf{H}_{T,k}$ to $\mathbf{H}_{S,k}$, while the second one is to construct each $\mathbf{G}_{T,k}$ by learning a nonlinear cross-domain feature mapping through a neural network.

Recall that $\overline{\mathbf{X}}_S = \left[\mathbf{X}_S\, \mathbf{X}_S^{(c)}\right]$ and $\overline{\mathbf{X}}_T = \left[\mathbf{X}_T\, \mathbf{X}_T^{(c)}\right]$. For simplicity in presentation, we denote by $\overline{\mathbf{H}}_{S,1} = \overline{\mathbf{X}}_S$ and $\overline{\mathbf{H}}_{T,1} = \overline{\mathbf{X}}_T$, respectively, where $\overline{\mathbf{H}}_{S,1} = \left[\mathbf{H}_{S,1}\, \mathbf{H}_{S,1}^{(c)}\right]$ and $\overline{\mathbf{H}}_{T,1} = \left[\mathbf{H}_{T,1}\, \mathbf{H}_{T,1}^{(c)}\right]$. We first recursively apply mSDA, i.e., (3) and (5) described in Section 2.2, on $\overline{\mathbf{H}}_{S,k}$ and $\overline{\mathbf{H}}_{T,k}$ to learn higher-level features $\overline{\mathbf{H}}_{S,k+1}$ and $\overline{\mathbf{H}}_{T,k+1}$, where $k = 1, 2, \ldots, K-1$. Note that when there is no bias on the corresponding instances in the target domain, one can simply set $\mathbf{W}_T$ in (3) to be the identity matrix of the dimensionality $d_T + 1$, and replace $\tanh(\cdot)$ in (5) by the identity function.

3.1.1. Approach I: constructing linear mapping for $\mathbf{G}_{T,k}$

In each layer k, we have the cross-domain correspondences represented by $\left\{\mathbf{H}_{S,k}^{(c)}, \mathbf{H}_{T,k}^{(c)}\right\}$. We aim to learn a linear feature transformation $\mathbf{G}_{T,k} \in \mathbb{R}^{(d_S+1)\times(d_T+1)}$ by solving the following minimization problem,

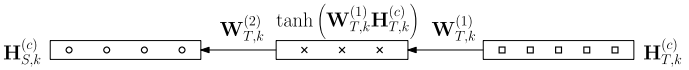


Fig. 5. A neural network for learning cross-domain feature map (Deep Architecture I).

$$\min_{\mathbf{G}_{T,k}} \left\| \mathbf{H}_{S,k}^{(c)} - \mathbf{G}_{T,k}\mathbf{H}_{T,k}^{(c)} \right\|_F^2 + \lambda \left\| \mathbf{G}_{T,k} \right\|_F^2, \tag{6}$$

where $\lambda > 0$ is a parameter of the regularization term on $\mathbf{G}_{T,k}$, which controls the tradeoff between the alignment of the heterogeneous features and the complexity of $\mathbf{G}_{T,k}$. The optimization problem (6) has a closed form solution that can be written as follows,

$$\mathbf{G}_{T,k} = \left(\mathbf{H}_{S,k}^{(c)}\mathbf{H}_{T,k}^{(c)\top}\right)\left(\mathbf{H}_{T,k}^{(c)}\mathbf{H}_{T,k}^{(c)\top} + \lambda\mathbf{I}\right)^{-1}, \tag{7}$$

where $\mathbf{I}$ is the identity matrix of the dimensionality $d_T + 1$. In the sequel, we denote by HHTL-I$_L$ the approach that is based on Deep Architecture I and learns linear transformations for cross-domain feature mappings $\{\mathbf{G}_{T,k}\}$'s. Note that HHTL-I$_L$ was proposed in our previous work [15].

3.1.2. Approach II: constructing neural networks for $\mathbf{G}_{T,k}$

One limitation of HHTL-I$_L$ is that learning a linear transformation for $\mathbf{G}_{T,k}$ may not be powerful enough to capture the relationships of heterogeneous features between the source domain and the target domain in each layer k. Therefore, we offer another approach denoted by HHTL-I$_N$, which learns $\mathbf{G}_{T,k}$ by constructing a neural network with one hidden layer as shown in Fig. 5. The overall procedure of HHTL-I$_N$ is similar to that of HHTL-I$_L$, but learns the pair of weight matrices $\left\{\mathbf{W}_{T,k}^{(1)}, \mathbf{W}_{T,k}^{(2)}\right\}$ to construct $\mathbf{G}_{T,k}$ in each layer k by solving the following minimization problem through backpropagation [17],

$$\min_{\mathbf{G}_{T,k}=\left\{\mathbf{W}_{T,k}^{(1)},\mathbf{W}_{T,k}^{(2)}\right\}} \left\| \mathbf{H}_{S,k}^{(c)} - \mathbf{W}_{T,k}^{(2)} \tanh\left(\mathbf{W}_{T,k}^{(1)}\mathbf{H}_{T,k}^{(c)}\right) \right\|_F^2. \tag{8}$$

After learning all high-level features and cross-domain feature mappings for each layer, we represent each source-domain labeled instance $\mathbf{x}_{S_i}$ by $\mathbf{z}_{S_i}$ that augments its original features with all the learned high-level features[2] as

$$\mathbf{z}_{S_i} = \left[\mathbf{h}_{S_i,1}^\top \cdots \mathbf{h}_{S_i,K}^\top\right]^\top, \tag{9}$$

where $\mathbf{h}_{S_i,k}$ denotes the high-level feature representation of the instance $\mathbf{x}_{S_i}$ in the k-th layer, and $\mathbf{h}_{S_i,1} = \mathbf{x}_{S_i}$. We then apply a standard classification algorithm on $\{\mathbf{z}_{S_i}, y_{S_i}\}$'s to train a target classifier f. To make a prediction on a target domain instance $\mathbf{x}_T^*$, we first generate its high-level feature representations $\{\mathbf{h}_{T,k}^*\}_{k=1}^K$ based on the learned DNN$_T$, and represent it by

$$\mathbf{z}_T^* = \left[\left(\mathbf{G}_{T,1}\mathbf{h}_{T,1}^*\right)^\top \cdots \left(\mathbf{G}_{T,K}\mathbf{h}_{T,K}^*\right)^\top\right]^\top. \tag{10}$$

Finally, we apply the learned classifier f on $\mathbf{z}_T^*$ to make a prediction $f(\mathbf{z}_T^*)$. The reason why we augment different layers of features for both training and testing is because we aim to incorporate additional high-level features learned in different layers to alleviate the bias for both the source domain and the target domain without losing original feature information.

The overall algorithms of HHTL-I$_L$ and HHTL-I$_N$ are summarized in Algorithm 1.

3.2. Approaches based on Deep Architecture II

From Algorithm 1, it can be shown that both HHTL-I$_L$ and HHTL-I$_N$ are very efficient as the learning in DNN$_S$ and DNN$_T$ in Step 1 can be done independently and in parallel, and learning in cross-domain feature mappings in each layer in Step 2 can be parallelized as well afterwards. However, as we mentioned, for approaches developed based on the Deep Architecture I, the learning of the high-level features and the cross-domain feature mappings are not fully integrated with each other. As a result, they may fail to maximally boost the performance of knowledge transfer across heterogeneous domains. Therefore, we propose Deep Architecture II as shown in Fig. 4. Similar to Deep Architecture I, a pair of neural networks, DNN$_S$ and DNN$_T$, are learned on the source domain data and the target domain data, respectively. However, different from Deep Architecture I, in each layer k, a pair of cross-domain feature mappings, $\mathbf{G}_{T,k}$ and $\mathbf{G}_{S,k}$, is learned.

[2] In an ideal case, if the high-level features and the cross-domain heterogeneous feature mapping can be perfectly learned in the top layer, one can just use them for training a classifier. However, HHTL is a challenging problem. Though multiple layers are introduced, it is still very difficulty to learn high-level features and a cross-domain heterogeneous feature mapping in the top layer perfectly. Therefore, here we propose to perform feature augmentation of all the features and cross-domain feature mappings learned in each layer to represent heterogeneous domain instances.

Algorithm 1 HHTL based on Deep Architecture I (HHTL-I$_L$ and HHTL-I$_N$).

Input: Target domain unlabeled data $\mathbf{D}_T = \{\mathbf{x}_{T_i}\}_{i=1}^{n_1}$, source domain labeled data $\mathbf{D}_S = \{(\mathbf{x}_{S_i}, y_{S_i})\}_{i=1}^{n_2}$, cross-domain correspondences $\mathbf{D}_C = \left\{\left(\mathbf{x}_{S_i}^{(c)}, \mathbf{x}_{T_i}^{(c)}\right)\right\}_{i=1}^{n_c}$, a trade-off parameter λ, and the number of layers K.

Initializations: $\overline{\mathbf{H}}_{S,1} = \overline{\mathbf{X}}_S$ and $\overline{\mathbf{H}}_{T,1} = \overline{\mathbf{X}}_T$.

for $k = 1, ..., K-1$ **do**

1: Apply mSDA on $\overline{\mathbf{H}}_{S,k}$ and $\overline{\mathbf{H}}_{T,k}$ to generate high-level features $\overline{\mathbf{H}}_{S,k+1} = \text{mSDA}(\overline{\mathbf{H}}_{S,k})$ and $\overline{\mathbf{H}}_{T,k+1} = \text{mSDA}(\overline{\mathbf{H}}_{T,k})$, respectively.

end for

for $k = 1, ..., K$ **do**

2: Learn a cross-domain feature mapping $\mathbf{G}_{T,k}$ by solving (6) for HHTL-I$_L$ or (8) for HHTL-I$_N$.

end for

3: Do feature augmentation on source domain labeled data using (9), and train a classifier f with $\{\mathbf{z}_{S_i}, y_{S_i}\}$'s.

Output: f, $\{\mathbf{G}_{T,k}\}_{k=1}^{K}$, DNN$_S$ in terms of $\{\mathbf{W}_{S,k}\}_{k=1}^{K-1}$ and DNN$_T$ in terms of $\{\mathbf{W}_{T,k}\}_{k=1}^{K-1}$.

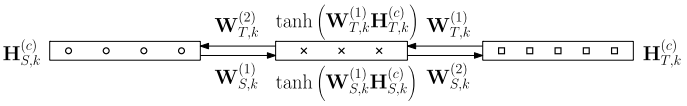


Fig. 6. A neural network for learning cross-domain feature map (Deep Architecture II).

In this section, we develop another family of three approaches based on Deep Architecture II. Similar to HHTL-I$_L$ and HHTL-I$_N$, in Deep Architecture II, the first approach is to construct each $\mathbf{G}_{T,k}$ (or $\mathbf{G}_{S,k}$) by learning a linear cross-domain feature mapping from $\mathbf{H}_{T,k}$ to $\mathbf{H}_{S,k}$ (or from $\mathbf{H}_{S,k}$ to $\mathbf{H}_{T,k}$), while the second one is to construct each $\mathbf{G}_{T,k}$ (or $\mathbf{G}_{S,k}$) by learning a nonlinear cross-domain feature mapping through a neural network. The third approach aims to train the cross-domain feature mappings $\mathbf{G}_{T,k}$ and $\mathbf{G}_{S,k}$ as well as the high-level features $\mathbf{H}_{T,k}$ and $\mathbf{H}_{S,k}$ at each layer jointly to further boost the performance of the deep learning model.

3.2.1. Approach I: constructing linear mapping for $\mathbf{G}_{T,k}$

Recall that $\overline{\mathbf{H}}_{S,k} = \left[\mathbf{H}_{S,k}\, \mathbf{H}_{S,k}^{(c)}\right]$ and $\overline{\mathbf{H}}_{T,k} = \left[\mathbf{H}_{T,k}\, \mathbf{H}_{T,k}^{(c)}\right]$. With the cross-domain correspondences $\left\{\mathbf{H}_{S,k}^{(c)}, \mathbf{H}_{T,k}^{(c)}\right\}$, we recursively learn a pair of linear feature transformations $\mathbf{G}_{T,k}$ and $\mathbf{G}_{S,k}$ by solving the following optimization problems, respectively,

$$\min_{\mathbf{G}_{T,k}} \left\|\mathbf{H}_{S,k}^{(c)} - \mathbf{G}_{T,k}\mathbf{H}_{T,k}^{(c)}\right\|_F^2 + \lambda \left\|\mathbf{G}_{T,k}\right\|_F^2, \tag{11}$$

$$\min_{\mathbf{G}_{S,k}} \left\|\mathbf{H}_{T,k}^{(c)} - \mathbf{G}_{S,k}\mathbf{H}_{S,k}^{(c)}\right\|_F^2 + \lambda \left\|\mathbf{G}_{S,k}\right\|_F^2. \tag{12}$$

Similar to solution (7), the closed-form solutions for above problems can be computed. Then we apply mSDA on $\left[\overline{\mathbf{H}}_{T,k}\, (\mathbf{G}_{S,k}\mathbf{H}_{S,k})\right]$ and $\left[\overline{\mathbf{H}}_{S,k}\, (\mathbf{G}_{T,k}\mathbf{H}_{T,k})\right]$ to learn high-level features $\overline{\mathbf{H}}_{T,k+1}$ and $\overline{\mathbf{H}}_{S,k+1}$, until all $\{\mathbf{G}_{T,k}\}$'s, $\{\mathbf{G}_{S,k}\}$'s, $\{\overline{\mathbf{H}}_{T,k}\}$'s, and $\{\overline{\mathbf{H}}_{S,k}\}$'s, where $k = 1, ..., K$, are obtained.

3.2.2. Approach II: constructing neural networks for $\mathbf{G}_k$

To capture nonlinear relationships of heterogeneous features between domains, we also construct neural networks with one hidden layer to approximate $\{\mathbf{G}_{T,k}\}$'s and $\{\mathbf{G}_{S,k}\}$'s, respectively, as shown in Fig. 6. Similar to HHTL-I$_N$, in each layer k, we construct one neural network for $\mathbf{G}_{T,k}$ (or $\mathbf{G}_{S,k}$) whose input layer represents $\mathbf{H}_{T,k}^{(c)}$ (or $\mathbf{H}_{S,k}^{(c)}$) and output layer represents $\mathbf{H}_{S,k}^{(c)}$ (or $\mathbf{H}_{T,k}^{(c)}$). Therefore, we can construct $\mathbf{G}_{T,k}$ and $\mathbf{G}_{S,k}$ by solving the following minimization problems, respectively,

$$\min_{\mathbf{G}_{T,k}=\left\{\mathbf{W}_{T,k}^{(1)},\mathbf{W}_{T,k}^{(2)}\right\}} \left\|\mathbf{H}_{S,k}^{(c)} - \mathbf{W}_{T,k}^{(2)} \tanh\left(\mathbf{W}_{T,k}^{(1)}\mathbf{H}_{T,k}^{(c)}\right)\right\|_F^2, \tag{13}$$

$$\min_{\mathbf{G}_{S,k}=\left\{\mathbf{W}_{S,k}^{(1)},\mathbf{W}_{S,k}^{(2)}\right\}} \left\|\mathbf{H}_{T,k}^{(c)} - \mathbf{W}_{S,k}^{(2)} \tanh\left(\mathbf{W}_{S,k}^{(1)}\mathbf{H}_{S,k}^{(c)}\right)\right\|_F^2, \tag{14}$$

whose solutions can be obtained using backpropagation and L-BFGS. In the sequel, we denote by HHTL-II$_N$ this approach. The overall algorithms of HHTL-II$_L$ HHTL-II$_N$ are summarized in Algorithm 2.

3.2.3. Approach III: constructing neural networks for co-learning $\mathbf{G}_k, \mathbf{H}_k$

The previous proposed methods use mSDA as the backbone to learn $\mathbf{H}_{T,k}$ and $\mathbf{H}_{S,k}$ for the target domain and the source domain. Unfortunately, due to the nature of mSDA, we cannot update the mappings in lower layers using the backpropagation algorithm. In this case, the feature learning of $\mathbf{H}_{S,k}$ is one-way and cannot be updated. In order to enable the automatic

Algorithm 2 HHTL based on Deep Architecture II (HHTL-II$_L$ and HHTL-II$_N$).

Input: Target domain unlabeled data $\mathbf{D}_T = \{\mathbf{x}_{T_i}\}_{i=1}^{n_1}$, source domain labeled data $\mathbf{D}_S = \{(\mathbf{x}_{S_i}, y_{S_i})\}_{i=1}^{n_2}$, cross-domain correspondences $\mathbf{D}_C = \left\{\left(\mathbf{x}_{S_i}^{(c)}, \mathbf{x}_{T_i}^{(c)}\right)\right\}_{i=1}^{n_c}$, a trade-off parameter λ, and the number of layers K.

Initializations: $\overline{\mathbf{H}}_{S,1} = \overline{\mathbf{X}}_S$ and $\overline{\mathbf{H}}_{T,1} = \overline{\mathbf{X}}_T$.

for $k = 1, ..., K-1$ **do**

1: Learn a pair cross-domain feature mappings $\mathbf{G}_{T,k}$ and $\mathbf{G}_{S,k}$ by solving (11) and (12) for HHTL-II$_L$ or (13) and (14) for HHTL-II$_N$.

2: Apply mSDA on $\left[\overline{\mathbf{H}}_{T,k}\ (\mathbf{G}_{S,k}\mathbf{H}_{S,k})\right]$ and $\left[\overline{\mathbf{H}}_{S,k}\ (\mathbf{G}_{T,k}\mathbf{H}_{T,k})\right]$ to learn high-level features $\overline{\mathbf{H}}_{T,k+1}$ and $\overline{\mathbf{H}}_{S,k+1}$, respectively,

$$\left[\overline{\mathbf{H}}_{T,k+1}\ \mathbf{U}_{T,k+1}\right] = \text{mSDA}\left(\left[\overline{\mathbf{H}}_{T,k}\ (\mathbf{G}_{S,k}\mathbf{H}_{S,k})\right]\right),$$

$$\left[\overline{\mathbf{H}}_{S,k+1}; \mathbf{U}_{S,k+1}\right] = \text{mSDA}\left(\left[\overline{\mathbf{H}}_{S,k}\ (\mathbf{G}_{T,k}\mathbf{H}_{T,k})\right]\right).$$

end for

3: Learn a pair of cross-domain feature mappings $\mathbf{G}_{T,K}$ and $\mathbf{G}_{S,K}$ for top layer by solving (11) and (12) for HHTL-II$_L$, or (13) and (14) for HHTL-II$_N$.

4: Do feature augmentation on source domain labeled data using (9), and train a classifier f with $\{\mathbf{z}_{S_i}, y_{S_i}\}$'s.

Output: f, $\{\mathbf{G}_{T,k}\}_{k=1}^{K}$, DNN$_S$ in terms of $\{\mathbf{W}_{S,k}\}_{k=1}^{K-1}$ and DNN$_T$ in terms of $\{\mathbf{W}_{T,k}\}_{k=1}^{K-1}$.

feature update in lower layers, we first adopt the second-order Taylor expansion and approximation for the mSDA objective. Based on this approximation, we further incorporate it into deep architecture II to achieve co-learning for $\mathbf{G}_k$ and $\mathbf{H}_k$. We named this approach as HHTL-II$_{CO}$, which is elaborated as follows.

The task of mSDA is to minimize the expected average loss under the corruption distribution $p(\tilde{\mathbf{x}}|\mathbf{x})$, as copies of the corruption $m \to \infty$, which can be equally expressed as follows,

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{p(\tilde{\mathbf{x}}_i|\mathbf{x}_i)}[\ell(\mathbf{x}_i, f_\theta(\tilde{\mathbf{x}}_i))], \tag{15}$$

where $f_\theta(\cdot)$ denotes the parameterized network to be optimized and loss function ℓ denotes the auto-encoder loss function and n denotes the number of training examples. In the sequel of analysis, we take one instance $\mathbf{x} \in \mathbb{R}^d$ and drop the subscript i for the simplicity. We approximate $\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}}))$ by its second-order Taylor expansion at the mean of corruption $\mu_{\mathbf{x}} = \mathbb{E}_{p(\tilde{\mathbf{x}})|\mathbf{x}}$ as follows,

$$\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}})) \approx \ell(\mathbf{x}, f_\theta(\mathbf{x})) + (\tilde{\mathbf{x}} - \mu_x)^\top \nabla_{\tilde{\mathbf{x}}}\ell + \frac{1}{2}(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})^\top \nabla^2_{\tilde{\mathbf{x}}}\ell(\tilde{\mathbf{x}} - \mu_{\mathbf{x}}), \tag{16}$$

where $\nabla_{\tilde{\mathbf{x}}}\ell$ and $\nabla^2_{\tilde{\mathbf{x}}}\ell$ are the first-order and second-order derivatives, respectively.

Taking the expectation of (16), we get the following approximation,

$$\mathbb{E}[\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}}))] \approx \ell(\mathbf{x}, f_\theta(\mathbf{x})) + \frac{1}{2}\text{tr}\left(\Sigma_{\mathbf{x}}\nabla^2_{\tilde{\mathbf{x}}}\ell\right), \tag{17}$$

where $\Sigma_{\mathbf{x}} = \mathbb{E}[(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})^\top]$ is the corruption variance matrix. Note that the above formulation only requires the first-order and the second-order statistics of the corrupted data. While this approximation could in principle be used to formulate our new learning algorithm, we conduct a few more computationally convenient simplifications to backpropagate second derivatives in (17).

Specifically, the corruption is applied to each dimension of $\mathbf{x}$ independently, which immediately simplifies $\Sigma_{\mathbf{x}}$ to a diagonal matrix. It further implies that only diagonal entries of the Hessian $\nabla^2_{\tilde{\mathbf{x}}}\ell$ are computed. The z-th dimension of the Hessian's diagonal is computed through the chain rule as follows,

$$\frac{\partial^2\ell}{\partial\tilde{x}_z^2} = \left(\frac{\partial\mathbf{h}}{\partial\tilde{x}_z}\right)^\top \frac{\partial^2\ell}{\partial\mathbf{h}^2}\frac{\partial\mathbf{h}}{\partial\tilde{x}_z} + \left(\frac{\partial\ell}{\partial\mathbf{h}}\right)^\top \frac{\partial^2\mathbf{h}}{\partial\tilde{x}_z^2}, \tag{18}$$

where $\mathbf{h}$ is the latent representation. Suggested by [18,19], the last term could be dropped to facilitate computation, and thus the right-hand side of (17) could be reduced to the following form,

$$\ell(\mathbf{x}, f_\theta(\mathbf{x})) + \frac{1}{2}\sum_{z=1}^{d}\sigma^2_{\mathbf{x},z}\sum_{v=1}^{d_h}\frac{\partial^2\ell}{\partial h_v^2}\left(\frac{\partial h_v}{\partial\tilde{x}_z}\right)^2, \tag{19}$$

where $\sigma^2_{\mathbf{x},z}$ is the z-th element of $\Sigma_{\mathbf{x}}$'s diagonal and h_v is the v-th element in the hidden layer representation $\mathbf{h} \in \mathbb{R}^{d_h}$. If we use the random corruption strategy in mSDA, then $\mu_{\mathbf{x}} = \mathbf{x}, \sigma^2_{\mathbf{x},z} = x_z^2 p/(1-p)$ with corruption probability p. For the multiple hidden layers, we can apply the aforementioned strategy to get the corresponding approximate second-order derivatives. For example, for the k-th hidden layer $\mathbf{h}_k$, the u-th element $h_{k,u}$ in its second-order derivative can be approximated as follows,

$$\frac{\partial^2\ell}{\partial h_{k,u}^2} \approx \sum_v \frac{\partial^2\ell}{\partial h_{k+1,v}^2}\left(\frac{\partial h_{k+1,v}}{\partial h_{k,u}}\right)^2, \tag{20}$$

where $h_{k+1,v}$ is v-th element in the $(k+1)$-th hidden layer representation $\mathbf{h}_{k+1}$. To this end, we can further update deep model parameters on the source and the target domains through optimizing the following objective function respectively,

$$\ell_S(\mathbf{x}_S) = \ell(\mathbf{x}_S, f_{\theta_S}(\mathbf{x}_S)) + \frac{1}{2}\sum_{z=1}^{d_S}\sigma^2_{\mathbf{x}_{S,z}}\sum_{v=1}^{d_{S,h}}\frac{\partial^2\ell}{\partial h^2_{S,v}}\left(\frac{\partial h_{S,v}}{\partial \tilde{x}_{S,z}}\right)^2, \tag{21}$$

$$\ell_T(\mathbf{x}_T) = \ell(\mathbf{x}_T, f_{\theta_T}(\mathbf{x}_T)) + \frac{1}{2}\sum_{z=1}^{d_T}\sigma^2_{\mathbf{x}_{T,z}}\sum_{v=1}^{d_{T,h}}\frac{\partial^2\ell}{\partial h^2_{T,v}}\left(\frac{\partial h_{T,v}}{\partial \tilde{x}_{T,z}}\right)^2, \tag{22}$$

where θ_S and θ_T denote the collection for parameters of DNN_S and DNN_T on the source and the target domains respectively, i.e., $\theta_S = \{\mathbf{W}_{S,k}\}_{k=1}^K, \theta_T = \{\mathbf{W}_{T,k}\}_{k=1}^K$. Recall that the cross-domain feature learning objectives are

$$g_{T,k}(\mathbf{h}_T^{(c)}, \mathbf{h}_S^{(c)}) = \left\|\mathbf{h}_{S,k}^{(c)} - \mathbf{W}_{T,k}^{(2)}\tanh\left(\mathbf{W}_{T,k}^{(1)}\mathbf{h}_{T,k}^{(c)}\right)\right\|_F^2, \tag{23}$$

$$g_{S,k}(\mathbf{h}_T^{(c)}, \mathbf{h}_S^{(c)}) = \left\|\mathbf{h}_{T,k}^{(c)} - \mathbf{W}_{S,k}^{(2)}\tanh\left(\mathbf{W}_{S,k}^{(1)}\mathbf{h}_{S,k}^{(c)}\right)\right\|_F^2. \tag{24}$$

By incorporating the above cross-domain feature learning objectives into the domain feature learning loss, we arrive at the following unified objective,

$$\ell_{overall} = \sum_{i=1}^{n_2}\ell_S(\mathbf{x}_{S_i}) + \sum_{i=1}^{n_1}\ell_T(\mathbf{x}_{T_i}) + \sum_{i=1}^{n_c}\sum_{k=1}^{K}\left(g_{T,k}(\mathbf{h}_{T_i}^{(c)}, \mathbf{h}_{S_i}^{(c)}) + g_{S,k}(\mathbf{h}_{T_i}^{(c)}, \mathbf{h}_{S_i}^{(c)})\right). \tag{25}$$

The model with the objective (25) (termed Co-learning Network) is able to jointly learn the in/cross-domain features and be trained by SGD, which is easily implemented by using deep learning packages such as Tensorflow [20] and Keras [21]. The algorithm is summarized in Algorithm 3.

Algorithm 3 HHTL-II based co-learning network (HHTL-II$_{CO}$).

Input: Target domain unlabeled data $\mathbf{D}_T = \{\mathbf{x}_{T_i}\}_{i=1}^{n_1}$, source domain labeled data $\mathbf{D}_S = \{(\mathbf{x}_{S_i}, y_{S_i})\}_{i=1}^{n_2}$, cross-domain correspondences $\mathbf{D}_C = \left\{\left(\mathbf{x}_{S_i}^{(c)}, \mathbf{x}_{T_i}^{(c)}\right)\right\}_{i=1}^{n_c}$, a trade-off parameter λ, and the number of layers K.
Initializations: Randomly Initialize all the network parameters.
Updating Parameters: Update in-domain network parameters θ_S, θ_T and cross-domain feature mappings $\{\mathbf{G}_{S,k}, \mathbf{G}_{T,k}\}_{k=1}^K$ by solving (25) with Backpropagation.
Feature Augmentation: Do feature augmentation on source domain labeled data using (9), and train a classifier f with $\{\mathbf{z}_{S_i}, y_{S_i}\}$'s.
Output: f, $\{\mathbf{G}_{T,k}\}_{k=1}^K$, DNN_S in terms of θ_S and DNN_T in terms of θ_T.

4. Experiments

In experiments, we verify the effectiveness of the proposed HHTL framework on a number of cross-language classification tasks compared with several baseline methods in terms of classification accuracy, and explore the impact of the number of layers in the proposed deep architectures, and parameter sensitivity.

4.1. Experimental setup

Sentiment Analysis: The cross-language sentiment dataset [22] comprises of Amazon product reviews of three product categories: books, DVDs and music. These reviews are written in four languages: English (EN), German (GE), French (FR), and Japanese (JP). For each language, the reviews are split into a train file and a test file, including 2,000 reviews per category. We use the English reviews in the train file as the source-domain labeled data (or target-domain unlabeled data), the non-English (each of the other three languages) reviews in the train file as the target-domain unlabeled data (or source-domain labeled data). Moreover, we apply Google translator on the non-English reviews in the test file to construct the cross-domain (English v.s. non-English) unlabeled correspondences.
Topic Categorization: The multilingual Reuters collection is a text dataset with 5,000 news articles from six topics (i.e., C15, CCAT, E21, ECAT, GCAT and M11) in five languages, English (EN), French (FR), German (GE), Italian (IT) and Spanish (SP), which are represented by a bag-of-words representation weighted by TF-IDF. Each document is also translated into the other four languages to construct correspondences in this dataset. Similar to the cross-language sentiment dataset, because in practice English documents are widely accessible, we take English as the source domain (or target domain), and each of the other languages as a target domain (or source domain), respectively. The performance of all methods are evaluated on the target-domain unlabeled data without any target-domain labeled data for training.

Baseline Methods: Because in our HTL setting, no labeled data is available in training, existing HTL methods that require target-domain labeled data cannot be used as baselines for comparison. Therefore, we compare the proposed HHTL framework with the following baseline methods and their deep learning extension where we used mSDA as the backbone for feature learning:

- **SVM-SC/mSDA-SVM-SC**: We design a baseline based on translation. We first train a classifier on the source-domain labeled data, and then make predictions on the source-domain corresponding data. In this way, the predicted labels on the source-domain corresponding data can be transferred to the target-domain corresponding data (translations). Finally, we train a target classifier with the "labeled" target-domain corresponding data to make predictions on the target-domain test data.
- **CL-KCCA/mSDA-KCCA**: We apply Cross-Lingual Kernel Canonical Component Analysis (CL-KCCA) [23] on the unlabeled correspondences between domains to learn two projections for the source and target languages, and then train a monolingual classifier with the projected source-domain labeled data in the common latent space. Testing on target-domain unlabeled data is performed in the latent space as well.
- **HeMap/mSDA-HeMap**: We apply heterogeneous Spectral Mapping (HeMap) [24] to learn mappings to project data from both domains onto a common feature subspace. Note that HeMap does not take the instance correspondence information into consideration.
- **TSL/mSDA-TSL**: We apply the correspondence-based HTL method proposed in [25], where the HTL problem is transformed into a standard matrix completion (MC) problem, to train a classifier to predict the unlabeled data in the target domain.
- **DAMA/mSDA-DAMA**: We apply Domain Adaptation using Manifold Alignment (DAMA) [26] to learn a common feature subspace by utilizing the cross-domain correspondences for the manifold alignment.

For all experiments, we employ the linear support vector machine (SVM) [27] with default parameter settings as the base classifier. We use the cross-validation method to adjust the model parameters. Specifically, we choose λ in the range of $\{0.01, 0.1, 1, 10, 100\}$ for HHTL, choose corruption probability p in mSDA in the range of $\{0.5, 0.6, 0.7, 0.8, 0.9\}$. By considering the cost of memory and computation, we fix the number of layers K used in DNN_S and DNN_T to be 3 when comparing HHTL with other baseline methods.[3] We tune the parameter κ for CL-KCCA (see (5) in [23]), the parameter β for HeMap (see (1) in [24]), the parameter γ for TSL (see (2) in [25]), the parameter μ for DAMA (see Theorem 1 in [26]) in the range of $\{0.01, 0.1, 1, 10, 100\}$. For a fair comparison, we set the number of layers of all mSDA based baselines to be 3 as well.

4.2. Performance comparison

We evaluate the performance of all the comparison methods under two learning settings: 1) cross-language, and 2) cross-language + cross-product.

4.2.1. Comparison results in the cross-language setting

In this setting, the bias issue is caused by language translation as described in the first motivating example in Section 1. To conduct comparison experiments in this setting, we generate a number of cross-lingual classification tasks. For the tasks with English as the source domain on the cross-language sentiment dataset, the original English reviews on all the three products are used as the source-domain labeled data, and non-English reviews on all the three products are considered as the target-domain data during the training process. From the target domain data, we randomly choose 2,000 non-English reviews, and translate them to English to form the correspondences between domains. The remaining non-English reviews are used as the target-domain unlabeled data. The constructions on the tasks with English as the target domain are similar. The averaged results in terms of accuracy with standard deviation on the target-domain unlabeled data over 10 random runs are reported in Table 1, where, for instance, EN-FR denotes that English is used as the source domain while French is used as the target domain.

Similarly, for the tasks with English as the source domain on the multilingual Reuters collection, we used the English documents on all the six topics as the source-domain labeled data, and the non-English reviews on all the six topics are considered as the target domain data. From the target domain data, we randomly choose 5,000 non-English documents, and translate them to English to form the correspondence between domains. The remaining non-English documents are used the target-domain unlabeled data. The constructions on the tasks with English as the target domain are similar. The averaged results in terms of accuracy on the reviews of each non-English language over 10 random runs are reported in Table 2.

[3] Note that regarding neural networks to learn cross-domain feature mappings in HHTL-I_N, HHTL-II_N and HHTL-II_{CO}, we only use one hidden layer for all experiments.

Table 1
Cross-language setting: sentiment classification (averaged acc ± std in %).

EN-FR	SVM-SC 73.10 ± 0.63	CL-KCCA 75.50 ± 1.54	HeMap 50.23 ± 1.26	DAMA 74.36 ± 1.54	TSL 73.27 ± 1.07
	mSDA-SVM-SC 77.03 ± 0.80	mSDA-KCCA 80.80 ± 2.05	mSDA-HeMap 70.45 ± 1.20	mSDA-DAMA 81.16 ± 1.82	mSDA-TSL 79.28 ± 1.50
	HHTL-I_L 82.67 ± 1.68	HHTL-I_N 82.87 ± 1.21	HHTL-II_L 83.26 ± 1.50	HHTL-II_N 84.12 ± 1.83	HHTL-CO **86.20 ± 1.59**
EN-GE	SVM-SC 73.05 ± 1.02	CL-KCCA 75.00 ± 1.40	HeMap 49.83 ± 1.08	DAMA 75.89 ± 2.20	TSL 74.55 ± 1.25
	mSDA-SVM-SC 77.01 ± 1.05	mSDA-KCCA 78.25 ± 1.57	mSDA-HeMap 70.52 ± 1.52	mSDA-DAMA 80.40 ± 1.36	mSDA-TSL 76.67 ± 1.42
	HHTL-I_L 82.76 ± 2.20	HHTL-I_N 83.10 ± 1.03	HHTL-II_L 83.25 ± 1.42	HHTL-II_N 84.06± 1.39	HHTL-CO **87.30 ± 1.51**
EN-JP	SVM-SC 65.50 ± 0.77	CL-KCCA 66.82 ± 1.25	HeMap 51.30 ± 2.05	DAMA 68.32 ± 1.72	TSL 67.82 ± 1.60
	mSDA-SVM-SC 71.03 ± 1.10	mSDA-KCCA 70.19 ± 2.20	mSDA-HeMap 60.40 ± 1.03	mSDA-DAMA 72.68 ± 1.25	mSDA-TSL 70.90 ± 1.44
	HHTL-I_L 75.95 ± 1.58	HHTL-I_N 76.12 ± 1.50	HHTL-II_L 76.71 ± 1.55	HHTL-II_N 77.40± 1.11	HHTL-CO **79.20 ± 1.52**
FR-EN	SVM-SC 69.82 ± 0.98	CL-KCCA 71.45 ± 2.13	HeMap 48.89 ± 1.37	DAMA 72.24 ± 1.63	TSL 70.73 ± 1.12
	mSDA-SVM-SC 71.80 ± 1.50	mSDA-KCCA 72.15 ± 1.84	mSDA-HeMap 62.47 ± 1.60	mSDA-DAMA 73.00 ± 1.30	mSDA-TSL 72.05 ± 1.65
	HHTL-I_L 72.15 ± 1.84	HHTL-I_N 79.67 ± 1.43	HHTL-II_L 80.55 ± 1.52	HHTL-II_N 81.77 ± 1.20	HHTL-II_{CO} **83.26 ± 1.68**
GE-EN	SVM-SC 68.95 ± 1.45	CL-KCCA 70.24 ± 1.80	HeMap 50.45 ± 1.62	DAMA 71.67 ± 1.59	TSL 72.07 ± 1.47
	mSDA-SVM-SC 72.82 ± 1.60	mSDA-KCCA 73.80 ± 1.56	mSDA-HeMap 60.50 ± 1.05	mSDA-DAMA 74.50 ± 1.75	mSDA-TSL 73.00 ± 1.39
	HHTL-I_L 77.76 ± 1.42	HHTL-I_N 78.26 ± 1.86	HHTL-II_L 80.02 ± 1.50	HHTL-II_N 81.31± 1.42	HHTL-II_{CO} **83.10 ± 1.55**
JP-EN	SVM-SC 67.00 ± 1.47	CL-KCCA 68.27 ± 1.11	HeMap 50.15 ± 1.20	DAMA 69.58 ± 1.67	TSL 68.16 ± 1.52
	mSDA-SVM-SC 70.95 ± 1.64	mSDA-KCCA 71.45 ± 2.62	mSDA-HeMap 60.30 ± 1.38	mSDA-DAMA 73.08 ± 1.51	mSDA-TSL 71.50 ± 1.80
	HHTL-I_L 74.06 ± 1.72	HHTL-I_N 76.86 ± 1.46	HHTL-II_L 77.89 ± 1.52	HHTL-II_N 78.67± 1.63	HHTL-II_{CO} **80.02 ± 1.75**

From Tables 1 and 2, we observe that the proposed HHTL approaches (HHTL-I_L, HHTL-I_N, HHTL-II_L, HHTL-II_N and HHTL-II_{CO}) outperform all the other baseline methods significantly in terms of classification accuracy. We also observe that the baseline equipped with mSDA can also largely improve their performance. The performance of CL-KCCA, DAMA, TSL and SVM-SC is much better than HeMap. The inferior performance of HeMap is caused by the fact that HeMap discards the valuable corresponding information in training. The cross-domain correspondences are incorporated either in a naive way (SVM-SC), dimension reduction (KCCA), manifold alignment (DAMA), matrix completion (TSL). Overall, these methods have comparable performance. However, mSDA-CCA performs slightly better than CL-KCCA, DAMA, TSL, and much better than all the other baselines.

Intuitively, the quality of the cross-domain heterogeneous feature mapping depends on its construction method. As shown in Tables 1 and 2, HHTL-I_N and HHTL-II_N outperform HHTL-I_L and HHTL-II_L, respectively. This suggests that nonlinear feature mappings constructed by neural networks can capture the complicated relationships between different languages more precisely than the linear feature mappings. Moreover, HHTL-II_{CO} performs the best, showing the benefits of joint optimization of the parameters.

4.2.2. Comparison results in the cross-language + cross-product setting

In this setting, we focus on cross-language + cross-product classification problems. To conduct comprehensive comparisons, we generate 18 cross-language and cross-product sentiment classification tasks with an English product domain as the source domain and 18 cross-language and cross-product sentiment classification tasks with an English product domain as the target domain on the cross-language sentiment dataset. For instance, the task EN-B-FR-D denotes that we use English Book reviews as the source-language labeled data, the French DVD reviews as the target-language test data, and all the French Book reviews in the test file and its English translations as the correspondences. Similarly, FR-B-EN-D denotes that we use French Book reviews as the source-language labeled data, the English DVD reviews as the target-language test data, and all the English Book reviews in the test file and its French translations as the correspondences.

The results are summarized in Tables 3 and 4. This setting is more challenging than the previous one due to cross problem and cross language shifts. We observe that the performance of all methods is lower than that in Tables 1 and 2, but

Table 2
Cross-language setting: topic categorization (averaged acc ± std in %).

EN-FR	SVM-SC 63.77 ± 1.25	CL-KCCA 63.41 ± 2.35	HeMap 51.07 ± 1.47	DAMA 64.89 ± 1.63	TSL 63.88 ± 2.05
	mSDA-SVM-SC 70.60 ± 1.47	mSDA-KCCA 71.53 ± 1.20	mSDA-HeMap 60.50 ± 1.59	mSDA-DAMA 73.08 ± 1.60	mSDA-TSL 72.10 ± 1.92
	HHTL-I_L 74.29 ± 1.23	HHTL-I_N 74.82 ± 1.30	HHTL-II_L 76.23 ± 1.17	HHTL-II_N 76.57 ± 2.21	HHTL-II_{CO} **78.35 ± 1.80**
EN-GE	SVM-SC 47.59 ± 1.44	CL-KCCA 55.71 ± 2.06	HeMap 49.85 ± 1.26	DAMA 57.32 ± 2.01	TSL 54.31 ± 1.71
	mSDA-SVM-SC 53.70 ± 1.56	mSDA-KCCA 56.43 ± 2.0	mSDA-HeMap 52.31 ± 1.50	mSDA-DAMA 56.20 ± 1.45	mSDA-TSL 54.80 ± 1.20
	HHTL-I_L 58.23 ± 1.15	HHTL-I_N 59.05 ± 1.14	HHTL-II_L 59.50 ± 1.56	HHTL-II_N 60.02 ± 1.87	HHTL-II_{CO} **63.52 ± 1.60**
EN-IT	SVM-SC 49.23 ± 1.56	CL-KCCA 56.17 ± 1.89	HeMap 52.09 ± 1.35	DAMA 58.01 ± 1.26	TSL 55.17 ± 1.27
	mSDA-SVM-SC 58.40 ± 1.82	mSDA-KCCA 60.15 ± 1.85	mSDA-HeMap 54.33 ± 1.60	mSDA-DAMA 62.85 ± 1.55	mSDA-TSL 60.02 ± 1.45
	HHTL-I_L 63.60 ± 0.73	HHTL-I_N 64.33 ± 1.28	HHTL-II_L 65.78 ± 1.63	HHTL-II_N 66.89 ± 1.63	HHTL-II_{CO} **68.70 ± 1.52**
EN-SP	SVM-SC 53.63 ± 1.22	CL-KCCA 56.69 ± 2.03	HeMap 50.23 ± 1.28	DAMA 57.07 ± 2.23	TSL 55.23 ± 2.20
	mSDA-SVM-SC 56.38 ± 1.65	mSDA-KCCA 59.01 ± 1.54	mSDA-HeMap 55.50 ± 1.72	mSDA-DAMA 61.30 ± 1.85	mSDA-TSL 57.02 ± 1.66
	HHTL-I_L 62.88 ± 1.25	HHTL-I_N 63.05 ± 0.98	HHTL-II_L 64.09 ± 1.40	HHTL-II_N 65.35 ± 1.52	HHTL-II_{CO} **68.20 ± 1.70**
FR-EN	SVM-SC 61.23 ± 1.46	CL-KCCA 62.56 ± 2.04	HeMap 50.32 ± 1.63	DAMA 63.68 ± 1.44	TSL 62.67 ± 2.22
	mSDA-SVM-SC 66.02 ± 1.09	mSDA-KCCA 68.26 ± 1.53	mSDA-HeMap 56.45 ± 1.59	mSDA-DAMA 71.72 ± 1.15	mSDA-TSL 67.51 ± 1.68
	HHTL-I_L 70.02 ± 1.15	HHTL-I_N 72.36 ± 1.51	HHTL-II_L 73.06 ± 1.31	HHTL-II_N 74.85 ± 2.06	HHTL-II_{CO} **76.20 ± 2.30**
GE-EN	SVM-SC 60.55 ± 1.28	CL-KCCA 61.07 ± 1.69	HeMap 49.66 ± 1.55	DAMA 62.54 ± 2.15	TSL 61.63 ± 1.50
	mSDA-SVM-SC 65.80 ± 1.23	mSDA-KCCA 66.08 ± 2.14	mSDA-HeMap 55.70 ± 1.59	mSDA-DAMA 68.35 ± 1.28	mSDA-TSL 65.47 ± 1.45
	HHTL-I_L 68.50 ± 1.08	HHTL-I_N 69.05 ± 1.52	HHTL-II_L 72.01 ± 1.47	HHTL-II_N 73.02 ± 1.53	HHTL-II_{CO} **77.08 ± 1.36**
IT-EN	SVM-SC 58.61 ± 1.49	CL-KCCA 59.86 ± 1.45	HeMap 50.43 ± 1.67	DAMA 60.86 ± 1.36	TSL 58.69 ± 1.63
	mSDA-SVM-SC 65.53 ± 1.60	mSDA-KCCA 66.63 ± 1.75	mSDA-HeMap 58.20 ± 1.70	mSDA-DAMA 68.45 ± 1.57	mSDA-TSL 65.47 ± 1.20
	HHTL-I_L 68.60 ± 1.55	HHTL-I_N 70.36 ± 1.56	HHTL-II_L 72.56 ± 1.42	HHTL-II_N 73.66 ± 1.86	HHTL-II_{CO} **76.83 ± 1.61**
SP-EN	SVM-SC 60.37 ± 1.61	CL-KCCA 60.45 ± 2.03	HeMap 50.06 ± 1.43	DAMA 62.96 ± 2.08	TSL 60.96 ± 1.58
	mSDA-SVM-SC 64.80 ± 1.90	mSDA-KCCA 64.97 ± 1.67	mSDA-HeMap 57.52 ± 1.86	mSDA-DAMA 67.60 ± 1.56	mSDA-TSL 63.85 ± 1.60
	HHTL-I_L 68.88 ± 1.25	HHTL-I_N 70.42 ± 1.05	HHTL-II_L 72.02 ± 1.32	HHTL-II_N 72.80 ± 1.75	HHTL-II_{CO} **75.32 ± 1.28**

the relations between methods are similar. In this setting the difference between HHTL-II and HHTL-I is more pronounced and, as before, the proposed HHTL-II_{CO} achieves the best performance on all the tasks.

4.3. Transfer distance

We also conduct experiment to analyze the distance between original instances in a domain and those "translated" from the other domain. Ben-David et al. [28] introduced the Proxy-A-distance (PAD) as a measure of how different two domains are from each other. The metric is defined as $2(1-2\epsilon)$, where ϵ is the generalization error of a classifier (a linear SVM in our case) trained on the binary classification problem to distinguish inputs between the two domains. PAD gives 0 when $\epsilon = 0.5$, i.e. random guessing, and in this case it is hard to distinguish the samples of two domains. PAD gives 2 when $\epsilon = 0$, i.e. no error, which means that is it easy to distinguish the samples of two domains. In this experiment, we choose the cross-language sentiment classification as a showcase to analyze the PAD of the source domain instances and the "translated" instances from the target domain. Table 5 summarizes the results of the PAD before and after HHTL approaches are applied, where the PAD before HTL corresponds HHTL with only one layer. From the results, we observe that HHTL approaches reduce the PAD compared to the original feature space, which means that the domain difference is reduced after applying HHTL.

Table 3
Cross-language + cross-domain setting: sentiment classification with EN as the source language (average acc ± std in %).

Task	mSDA-SVM-SC	mSDA-CL-KCCA	mSDA-HeMap	mSDA-DAMA	mSDA-TSL	HHTL-I_L	HHTL-I_N	HHTL-II_L	HHTL-II_N	HHTL-II_{CO}
EN-B-FR-D	75.67 ± 1.36	72.96 ± 1.72	55.30 ± 1.26	76.33 ± 1.50	73.42 ± 1.62	76.75 ± 1.73	76.82 ± 1.46	78.23 ± 1.62	79.41 ± 1.58	**81.25 ± 1.70**
EN-B-FR-M	67.50 ± 1.82	64.29 ± 1.29	54.81 ± 1.40	68.28 ± 1.27	65.03 ± 1.86	67.65 ± 1.54	68.05 ± 1.38	69.00 ± 1.33	70.26 ± 1.43	**73.45 ± 1.38**
EN-B-GE-D	75.33 ± 1.70	78.23 ± 2.16	56.47 ± 1.38	76.20 ± 1.82	76.95 ± 1.61	75.10 ± 1.27	75.69 ± 1.61	78.76 ± 1.30	80.22 ± 1.96	**83.50 ± 1.46**
EN-B-GE-M	66.32 ± 2.05	62.52 ± 1.63	57.58 ± 1.62	70.80 ± 1.56	64.28 ± 1.73	69.55 ± 1.29	71.08 ± 1.45	72.42 ± 1.29	74.25 ± 1.61	**77.02 ± 1.61**
EN-B-JP-D	72.72 ± 1.65	71.95 ± 1.86	53.80 ± 1.81	73.00 ± 1.74	71.69 ± 1.62	72.56 ± 1.52	72.67 ± 1.10	73.02 ± 1.18	74.46 ± 1.40	**76.53 ± 1.58**
EN-B-JP-M	56.48 ± 2.10	57.17 ± 1.90	52.51 ± 1.45	59.95 ± 1.87	56.56 ± 1.82	62.39 ± 1.07	63.01 ± 1.32	64.77 ± 1.40	66.03 ± 1.08	**68.10 ± 1.36**
EN-D-FR-B	75.50 ± 1.67	72.32 ± 2.31	54.59 ± 1.07	77.76 ± 1.82	75.52 ± 1.40	79.27 ± 1.63	79.85 ± 1.38	80.89 ± 1.64	81.27 ± 1.42	**83.82 ± 1.92**
EN-D-FR-M	73.05 ± 1.53	68.87 ± 1.77	52.82 ± 1.56	75.87 ± 1.60	72.70 ± 1.63	75.43 ± 2.06	76.12 ± 1.46	76.81 ± 1.61	78.60 ± 1.39	**80.33 ± 1.57**
EN-D-GE-B	78.09 ± 1.56	75.32 ± 1.67	56.98 ± 1.60	80.91 ± 1.80	76.86 ± 1.70	79.35 ± 1.41	80.27 ± 1.55	81.45 ± 1.30	82.32 ± 1.24	**84.21 ± 1.53**
EN-D-GE-M	74.95 ± 1.45	75.69 ± 1.63	56.26 ± 1.20	77.62 ± 1.97	75.08 ± 1.30	78.55 ± 1.66	79.13 ± 1.29	80.50 ± 1.63	80.14 ± 1.47	**82.62 ± 1.50**
EN-D-JP-B	74.21 ± 1.57	72.60 ± 1.39	52.85 ± 1.72	75.86 ± 1.66	73.06 ± 1.86	68.12 ± 1.76	70.16 ± 1.65	72.78 ± 1.67	74.53 ± 1.61	**76.37 ± 1.60**
EN-D-JP-M	65.10 ± 1.54	66.89 ± 1.34	52.78 ± 1.30	68.38 ± 1.62	67.05 ± 1.48	70.58 ± 1.75	71.24 ± 1.63	72.42 ± 1.78	73.96 ± 1.27	**75.49 ± 1.66**
EN-M-FR-B	78.16 ± 1.70	74.23 ± 1.63	56.40 ± 1.60	79.45 ± 1.80	76.50 ± 1.07	77.80 ± 2.36	78.01 ± 1.75	78.86 ± 1.15	79.17 ± 1.36	**80.80 ± 1.82**
EN-M-FR-D	75.39 ± 1.82	72.76 ± 1.56	55.07 ± 1.45	76.68 ± 1.02	73.52 ± 1.18	77.04 ± 1.37	80.95 ± 1.42	81.55 ± 1.65	79.25 ± 1.69	**81.33 ± 1.64**
EN-M-GE-B	78.21 ± 1.52	76.82 ± 1.47	53.58 ± 1.06	79.10 ± 1.84	76.92 ± 1.50	78.30 ± 1.37	80.96 ± 1.60	81.55 ± 1.73	83.18 ± 1.33	**85.61 ± 1.82**
EN-M-GE-D	80.60 ± 1.06	72.28 ± 1.3	57.43 ± 1.70	81.58 ± 1.63	79.39 ± 1.75	81.42 ± 1.26	81.66 ± 1.64	82.78 ± 1.40	83.36 ± 1.49	**85.60 ± 1.62**
EN-M-JP-B	70.37 ± 1.69	68.83 ± 1.72	56.05 ± 1.28	72.06 ± 1.52	70.35 ± 1.57	71.65 ± 1.56	71.87 ± 2.10	72.48 ± 1.87	72.59 ± 1.63	**75.23 ± 1.56**
EN-M-JP-D	73.58 ± 1.70	71.06 ± 1.96	56.80 ± 1.05	74.46 ± 1.58	72.04 ± 1.65	74.25 ± 1.75	74.39 ± 1.38	76.26 ± 1.57	77.06 ± 1.68	**79.68 ± 1.30**

Table 4
Cross-language + cross-domain setting: sentiment classification with EN as the target language (average acc ± std in %).

Task	mSDA-SVM-SC	mSDA-CL-KCCA	mSDA-HeMap	mSDA-DAMA	mSDA-TSL	HHTL-I_L	HHTL-I_N	HHTL-II_L	HHTL-II_N	HHTL-II_{CO}
FR-B-EN-D	70.54 ± 1.61	70.60 ± 1.36	55.62 ± 1.54	72.36 ± 1.67	70.68 ± 1.60	71.99 ± 1.85	73.67 ± 1.68	74.23 ± 1.55	77.63 ± 1.61	**79.40 ± 1.58**
FR-B-EN-M	62.53 ± 1.70	61.91 ± 1.25	54.59 ± 1.60	66.35 ± 1.50	63.40 ± 1.53	65.52 ± 1.33	66.85 ± 1.65	67.01 ± 1.37	68.60 ± 1.35	**70.82 ± 1.62**
GE-B-EN-D	72.81 ± 1.82	72.56 ± 2.03	56.70 ± 1.67	74.60 ± 1.63	72.05 ± 1.32	73.67 ± 1.20	74.38 ± 1.65	76.44 ± 1.28	78.52 ± 1.32	**81.27 ± 1.50**
GE-B-EN-M	64.80 ± 2.05	62.63 ± 1.5	52.52 ± 1.62	67.82 ± 2.01	63.05 ± 1.70	69.85 ± 1.74	71.26 ± 1.53	73.05 ± 1.72	73.95 ± 1.74	**75.58 ± 1.56**
JP-B-EN-D	73.48 ± 1.24	73.55 ± 1.6	54.05 ± 1.16	75.62 ± 1.85	72.91 ± 1.34	74.42 ± 1.60	75.02 ± 1.52	75.64 ± 1.66	77.42 ± 1.36	**79.05 ± 1.45**
JP-B-EN-M	56.04 ± 2.01	55.40 ± 1.88	53.60 ± 1.50	58.32 ± 1.35	56.02 ± 1.61	58.65 ± 1.23	62.20 ± 1.48	64.52 ± 1.40	65.67 ± 1.28	**68.03 ± 1.20**
FR-D-EN-B	73.22 ± 1.07	71.45 ± 1.55	49.48 ± 1.57	71.86 ± 1.54	70.62 ± 1.65	77.46 ± 1.58	79.10 ± 1.62	80.52 ± 1.58	81.02 ± 1.52	**83.48 ± 1.55**
FR-D-EN-M	72.06 ± 1.50	69.64 ± 1.82	55.21 ± 1.03	74.31 ± 1.80	70.28 ± 1.52	73.62 ± 2.25	74.24 ± 1.53	75.66 ± 1.52	76.42 ± 1.61	**79.16 ± 1.40**
GE-D-EN-B	79.17 ± 1.18	77.70 ± 1.62	53.06 ± 1.52	79.83 ± 2.02	78.62 ± 1.63	78.44 ± 1.52	80.36 ± 1.45	80.78 ± 1.62	81.46 ± 1.52	**83.70 ± 1.18**
GE-D-EN-M	76.03 ± 2.28	74.57 ± 1.89	54.26 ± 1.64	78.67 ± 1.27	75.29 ± 1.42	76.45 ± 1.72	78.20 ± 1.65	80.50 ± 1.57	80.62 ± 1.56	**81.29 ± 1.60**
JP-D-EN-B	71.29 ± 1.30	71.60 ± 1.37	56.50 ± 1.47	72.02 ± 2.04	70.35 ± 1.28	70.23 ± 1.55	71.68 ± 1.28	72.46 ± 1.70	73.50 ± 1.29	**74.79 ± 1.46**
JP-D-EN-M	65.02 ± 2.07	62.90 ± 2.02	53.48 ± 1.45	68.40± 2.13	63.42 ± 1.58	68.23 ± 1.38	69.40 ± 1.35	71.40 ± 1.82	72.64 ± 1.55	**74.58 ± 2.09**
FR-M-EN-B	75.38 ± 1.33	73.03 ± 1.57	55.02 ± 2.18	76.78 ± 1.80	74.25 ± 1.55	75.40 ± 2.06	76.16 ± 1.84	77.32 ± 1.42	78.26 ± 1.72	**80.37 ± 1.50**
FR-M-EN-D	74.48 ± 2.10	72.64 ± 1.70	56.85 ± 1.18	76.43 ± 2.05	74.48 ± 2.01	75.50 ± 1.43	77.20 ± 1.57	78.68 ± 1.29	79.01 ± 1.34	**81.65 ± 2.02**
GE-M-EN-B	77.61 ± 1.78	75.06 ± 1.52	53.08 ± 1.57	78.93 ± 1.24	74.35 ± 1.70	77.06 ± 1.65	78.67 ± 1.51	80.62 ± 1.82	81.55 ± 1.63	**81.55 ± 1.63**
GE-M-EN-D	77.72 ± 1.80	75.28 ± 1.56	56.25 ± 1.43	77.25 ± 2.00	74.30 ± 1.64	78.20 ± 1.62	79.82 ± 1.81	81.70 ± 1.37	82.46 ± 1.48	**85.30 ± 1.80**
JP-M-EN-B	69.02 ± 1.43	68.69 ± 1.50	57.04 ± 1.20	69.70 ± 1.06	66.23 ± 1.25	71.65 ± 1.56	71.87 ± 2.10	72.48 ± 1.87	72.59 ± 1.63	**73.90 ± 2.10**
JP-M-EN-D	75.40 ± 1.62	73.06 ± 1.96	53.41 ± 1.26	76.01 ± 1.56	74.91 ± 1.26	74.28 ± 1.64	75.52 ± 1.47	76.06 ± 1.07	76.52 ± 1.80	**78.27 ± 1.05**

Table 5
Proxy a-distances comparisons.

Domains	Before	HHTL-I_L	HHTL-I_N	HHTL-II_L	HHTL-II_N	HHTL-II_{CO}
EN-FR	1.85	1.81	1.80	1.79	1.77	1.75
EN-GE	1.83	1.80	1.78	1.78	1.77	1.73
EN-JP	1.98	1.93	1.92	1.91	1.90	1.86
FR-EN	1.87	1.85	1.82	1.81	1.80	1.77
GE-EN	1.88	1.84	1.83	1.82	1.80	1.78
JP-EN	1.96	1.94	1.91	1.90	1.89	1.85

Table 6
Cross-language setting: Deep Architecture I with varying number of layers on sentiment classification (average accuracy ± standard deviation in %).

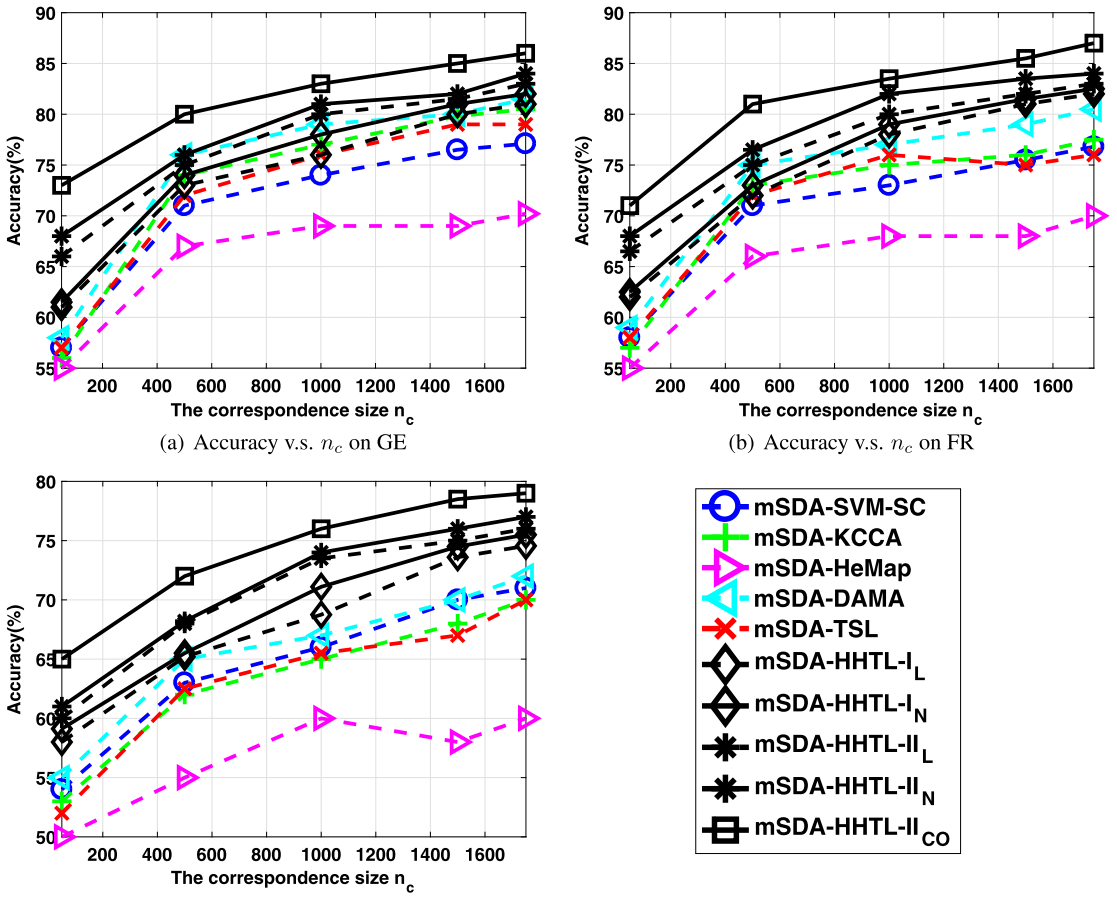
Task	HHTL-I_L (# layers)					
	1	2	3	4	5	6
EN-FR	74.01 ± 1.67	79.02 ± 1.50	82.67 ± 1.68	83.14 ± 1.75	83.40± 1.10	83.48± 1.03
EN-GE	73.08 ± 2.05	78.54 ± 1.87	82.76 ± 2.20	83.23 ± 2.21	84.00 ± 1.11	83.92 ± 1.20
EN-JP	66.12 ± 1.63	71.03 ± 2.01	75.95 ± 1.58	76.37 ± 1.63	76.78 ± 1.20	76.80 ± 1.01
FR-EN	70.01 ± 1.23	76.01 ± 1.45	79.67 ± 1.43	80.75 ± 1.85	81.03 ± 1.73	81.02 ± 1.05
GE-EN	67.65 ± 1.37	73.88 ± 1.96	77.76 ± 1.42	79.36 ± 1.23	80.21 ± 1.25	80.16 ± 1.10
JP-EN	67.00 ± 1.47	71.68 ± 2.33	74.06 ± 1.72	75.69 ± 1.76	76.10 ± 1.15	76.12 ± 1.23
	HHTL-I_N (# layers)					
	1	2	3	4	5	6
EN-FR	74.17 ± 1.62	80.79 ± 1.83	82.87 ± 1.21	84.54 ± 1.62	85.28 ± 1.62	85.26 ± 1.37
EN-GE	73.57 ± 1.84	79.45 ± 1.65	83.10 ± 1.03	85.04 ± 1.80	85.53 ± 1.37	85.56 ± 1.04
EN-JP	66.54 ± 1.70	72.46 ± 1.57	76.12 ± 1.50	78.46 ± 1.43	79.02 ± 1.14	79.01 ± 1.10
FR-EN	71.20 ± 2.03	77.38 ± 1.69	80.55 ± 1.52	82.01 ± 1.55	82.68 ± 1.05	82.70 ± 1.16
GE-EN	68.81 ± 1.63	75.33 ± 2.06	78.26 ± 1.86	80.22 ± 1.28	81.01 ± 1.13	81.00 ± 1.05
JP-EN	68.22 ± 1.85	72.40 ± 1.70	76.86 ± 1.46	77.69 ± 1.60	78.13 ± 1.40	78.20 ± 1.14

Table 7
Cross-language setting: Deep Architecture II with varying number of layers on sentiment classification with EN as the source language (average accuracy ± standard deviation in %).

Task	HHTL-II_L (# layers)					
	1	2	3	4	5	6
EN-FR	74.01 ± 1.67	80.56 ± 1.49	83.26 ± 1.50	84.05 ± 1.06	84.20 ± 1.02	84.23 ± 1.00
EN-GE	73.08 ± 2.05	79.42 ± 1.85	83.25 ± 1.42	84.71 ± 1.80	85.34 ± 1.67	85.40 ± 1.13
EN-JP	66.12 ± 1.63	72.15 ± 1.61	76.71 ± 1.55	77.54 ± 1.69	78.15 ± 1.65	78.10 ± 1.17
FR-EN	70.01 ± 1.23	76.33 ± 1.80	81.77 ± 1.20	82.35 ± 1.94	82.82 ± 1.57	82.85± 1.16
GE-EN	67.65 ± 1.37	75.67 ± 1.75	80.02 ± 1.50	82.12 ± 1.50	82.63 ± 1.40	82.60 ± 1.15
JP-EN	67.00 ± 1.47	73.34 ± 2.06	77.89 ± 1.52	80.06 ± 1.43	80.60 ± 1.15	80.71 ± 1.04
	HHTL-II_N (# layers)					
	1	2	3	4	5	6
EN-FR	74.17 ± 1.62	81.24 ± 2.04	84.12 ± 1.83	85.75 ± 1.75	86.21 ± 1.17	86.23 ± 1.10
EN-GE	73.57 ± 1.84	82.54 ± 1.85	84.06 ± 1.39	85.86 ± 1.80	86.10 ± 1.36	86.11 ± 1.15
EN-JP	66.54 ± 1.70	73.03 ± 1.47	77.40 ± 1.11	79.64 ± 1.56	80.05 ± 1.16	80.01 ± 1.03
FR-EN	71.20 ± 2.03	77.60 ± 1.69	82.26 ± 1.68	82.90 ± 1.42	83.27 ± 1.13	83.26 ± 1.06
GE-EN	68.81 ± 1.63	76.78 ± 1.68	81.31 ± 1.42	83.53 ± 1.50	84.16 ± 1.63	84.20 ± 1.12
JP-EN	68.22 ± 1.85	75.09 ± 1.55	78.67 ± 1.63	80.17 ± 1.41	80.60 ± 1.15	80.67 ± 1.11
	HHTL-II_{CO} (# layers)					
	1	2	3	4	5	6
EN-FR	78.31 ± 1.80	84.15 ± 1.72	86.20 ± 1.59	87.78 ± 1.59	89.50 ± 1.14	89.63 ± 1.26
EN-GE	78.02 ± 1.39	83.89 ± 1.81	87.30 ± 1.51	88.45 ± 1.63	88.93 ± 1.62	88.90 ± 1.13
EN-JP	70.14 ± 1.64	76.21 ± 1.50	79.20 ± 1.56	81.13 ± 1.17	81.40 ± 1.25	81.41 ± 1.18
FR-EN	76.56 ± 1.72	80.24 ± 1.58	82.26 ± 1.68	83.48 ± 1.60	83.75 ±1.52	83.90 ± 1.24
GE-EN	74.39 ± 1.60	79.06 ± 1.37	83.10 ± 1.55	84.60 ± 1.45	85.20 ± 1.46	85.18 ± 1.05
JP-EN	74.15 ± 1.58	78.38 ± 1.50	80.02 ± 1.75	82.30 ± 1.28	83.21 ± 1.53	83.10± 1.30

4.4. Impact of the number of layers in HHTL

In this experiment, we aim to analyze the impact of the number of layers used in HHTL. As shown in Tables 6 and 7. The more layers are used in HHTL, the better performance is achieved. This supports the intuition that high level features generated by deep layers reduce bias. On the other hand, the performance saturates when the layer size is larger than 5. As we mentioned, by considering the computational cost, we set the number of layers to 3 for HHTL to conduct experiments in previous sections.



(a) Accuracy v.s. n_c on GE

(b) Accuracy v.s. n_c on FR

(c) Accuracy v.s. n_c on JP

(d) Legend

Fig. 7. Parameter analysis.

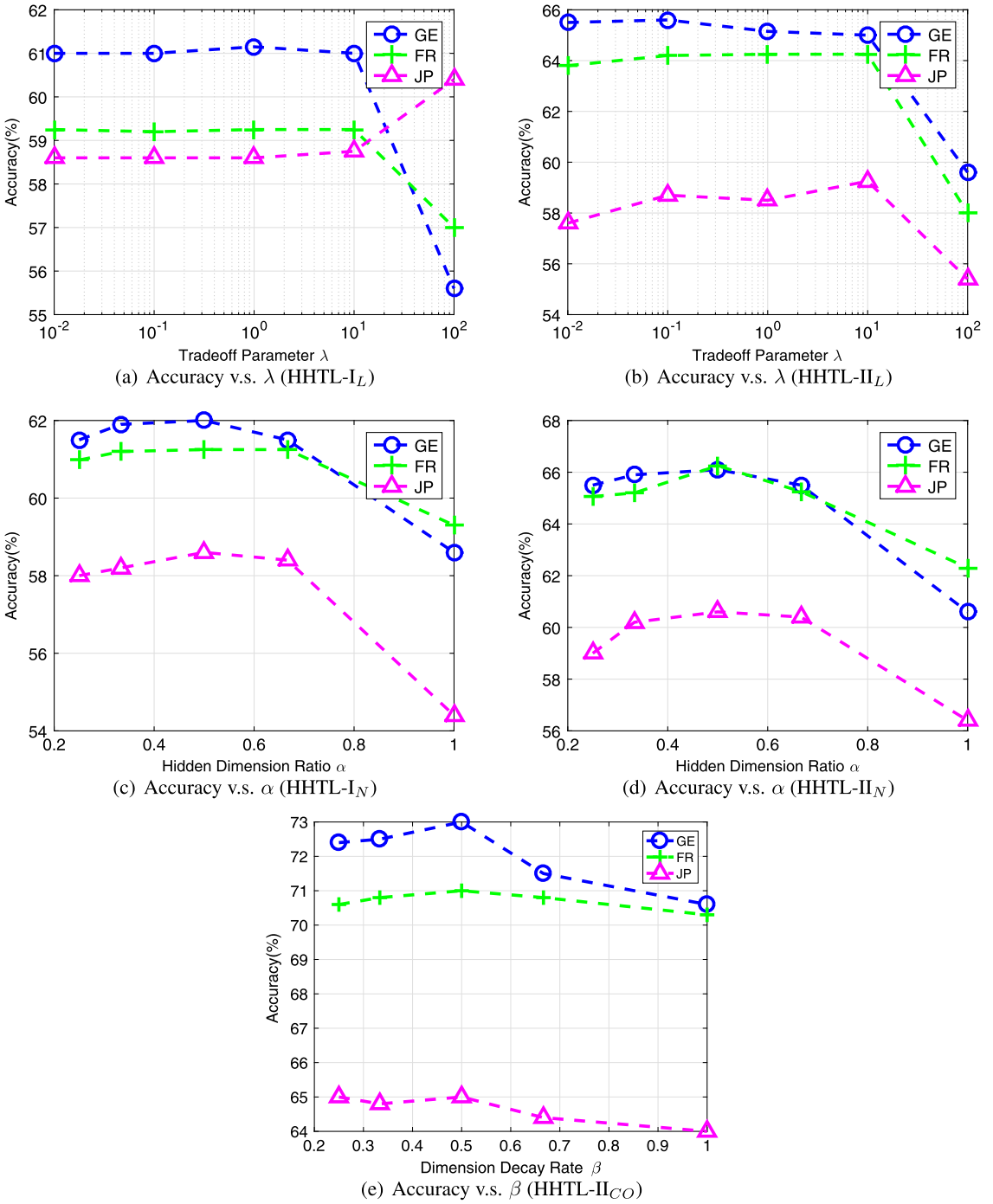
4.5. Impact of different correspondences size

In the HTL setting, a common assumption is that the number of correspondences across domains, n_c, is small, which may affect the performance of cross-language classification. In this section, we conduct experiments to analyze the impact of the correspondences size n_c to the overall performance of HHTL and the mSDA variant of baselines. Note that, here, we use the cross-language sentiment dataset under the cross-language setting to design experiments. We vary the correspondences size in the range of $[50, 250, 500, 750, 1000, 1250, 1500, 1750]$. The results are reported in Fig. 7. From the figures, we observe that all the methods that use the unlabeled correspondences consistently outperform mSDA-HeMap, which discards the correspondence information. We observe that all methods improve with larger correspondence sizes and the relative performance between methods is stable across sizes. The difference between HHTL-II and HHTL-I is more pronounced when the correspondence size is small. HHTL-II$_{CO}$ achieves the best performance on all correspondence sizes.

4.6. Parameter sensitivity study

In this section, we study the parameter sensitivity of the proposed HHTL approaches. Besides the number of layers and the size of cross-domain correspondences, both HHTL-I$_L$ and HHTL-II$_L$ have a tradeoff parameter λ, while HHTL-I$_N$ and HHTL-II$_N$ have an additional parameter specifying the size of the hidden layer dimension h of cross-domain neural networks. We first analyze how performance of HHTL-I$_L$ and HHTL-II$_L$ in terms of accuracy changes with varying values of λ in the range of $[0.001, 0.01, 1, 10, 100]$. From Figs. 8(a) and 8(b), we observe that the performance of both HHTL-I$_L$ and HHTL-II$_L$ is stable when λ is no more than 10.

We further analyze how the performance of HHTL-I$_N$ and HHTL-II$_N$ changes with varying values of h. To eliminate the effect of dimension scale, we use the ratio α between the hidden neurons size and the sum of source and target domain dimensions as measure of hidden layer size, i.e., $\alpha = \frac{h}{d_S + d_T}$. We vary the α in the range of $\left[\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, 1\right]$. From Figs. 8(c) and 8(d), we observe that the performance of both HHTL-I$_N$ and HHTL-II$_N$ in terms of accuracy remains stable when



(a) Accuracy v.s. λ (HHTL-I$_L$) (b) Accuracy v.s. λ (HHTL-II$_L$)

(c) Accuracy v.s. α (HHTL-I$_N$) (d) Accuracy v.s. α (HHTL-II$_N$)

(e) Accuracy v.s. β (HHTL-II$_{CO}$)

Fig. 8. Parameter sensitivity analysis of HHTL.

h is smaller than $d_s + d_T$. When h is around the mean of the source domain and target domain dimensions, HHTL-I$_N$ and HHTL-II$_N$ achieve their best performance, respectively. This observation also matches the empirically-derived rules-of-thumb in [29] and experimental results in [30]. Finally, we analyze the impact of the hidden neurons size of each domain in HHTL-II$_{CO}$ by fixing $\alpha = 1/2$. We define the dimension decay rate β to be the ratio between the size of the current layer and of the previous layer. i.e., $\beta = \frac{h_{K+1}}{h_K}$. We vary the β in the range of $\left[\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, 1\right]$. The results are shown in the Fig. 8(e). We observe that the performance achieves stable performance with decay rate $\beta < 1$. The smaller decay rate always leads satisfactory performance agreeing with the intuition that high level feature are usually more compact and requires fewer neurons.

5. Related work

Our proposed framework HHTL is mainly related to the following topics in machine learning: homogeneous transfer learning, heterogeneous transfer learning, and deep learning. In this section, we review related work in these topics.

5.1. Homogeneous transfer learning

Homogeneous transfer learning aims to improve generalization of a predictive model across different domains of the same feature space. Learning a good feature representation for different domain data is crucial for homogeneous transfer learning [28,1]. For instance, Blitzer et al. [4] proposed the structural correspondence learning algorithm (SCL) that uses the co-called "pivot" features across domains as a bridge to learn new features for reducing the domain difference. Pan et al. [5,31,32] proposed a series of dimensionality reduction methods based on Hilbert space embedding of distributions [33] to learn a low-dimensional space for both the source domain and the target domain data, where the distance in distributions between domains can be reduced while important properties of the original data, e.g., data variance or geometric structure, can be preserved. Daumé III [34] proposed a simple mapping function which augments the feature from both the source domain and the target domain in a high dimensional feature space. Daumé III et al. [35] further proposed an extension in a semi-supervised learning manner, where unlabeled data in the target domain are taken into consideration in learning. Gong et al. [36] proposed a kernel-based method to embed both the source domain and the target domain datasets into Grassmann manifolds and construct geodesic flows between them to model domain shift and learn new feature representations for the target domain.

5.2. Heterogeneous transfer learning

Heterogeneous transfer learning (HTL) aims to transfer knowledge across different feature spaces. A crucial research issue in HTL is to find a common feature representation for both the source domain and the target domain data, on which knowledge transfer is effective. In general, there are two approaches to learning a common representation for data of heterogeneous domains. One approach is to learn a pair of feature mappings to transform the source domain and the target domain data to a latent common feature space, respectively [24,22,26,37]. For instance, Shi et al. [24] proposed the Heterogenous Spectral Mapping method (HeMap) to learn a pair of feature mappings based on spectral embedding, where label information is discarded in learning. Wang and Mahadevan [26] proposed a manifold alignment method denoted by DAMA, to align heterogenous features in a latent space based on a manifold regularization term. In DAMA, label information in both the source domain and the target domain is exploited to construct a similarity matrix for manifold alignment. Duan et al. [37] proposed the Heterogenous Feature Augmentation method (HFA) to augment homogeneous common features learned by a SVM-style approach with heterogeneous features of the source domain and the target domain.

Another approach is to learn an asymmetric transformation to map data from one domain to another domain directly [6, 38]. Our proposed framework belongs to this approach. Kulis [6] proposed an Asymmetric Regularized Cross-domain transformation method (ARC-t) to learn an asymmetric transformation across domains based on metric learning. In ARC-t, label information in the both the source domain and the target domain is utilized to construct similarity and dissimilarity constraints between instances from the source domain and the target domain, respectively. The formulated metric learning problem can be solved by an alternating optimization algorithm. Zhou et al. [38] proposed to learn a sparse feature mapping between the source domain and the target domain by exploiting commonality between multiple binary classification tasks decomposed from the target multi-class classification problem.

Based on different assumptions on inputs for training, previous HTL approaches can be further classified into two settings. In a first setting, a few target-domain labeled data are assumed to be available for training [6,26,37,38], while in a second setting, some unlabeled correspondences between heterogeneous domains are assumed to be available for training [7,25, 39]. In the latter setting, which is our focus, Dai et al. [7] proposed a probabilistic model to construct a "translator" to build connections between instances from different domains. Xiao and Guo et al. [25] applied an existing matrix completion technique to HTL. Specifically, in their proposed method, with sufficient cross-domain correspondences given in advance, the goal is to reconstruct "missing correspondences" for all the instances observed in either the source domain or the target domain using matrix completion. Pan et al. [39] proposed a matrix-factorization-based approach to transfer knowledge across different recommender systems with heterogeneous user feedbacks by using some common users and items as a bridge.

However, most of these correspondence-based HTL methods implicitly assume that the cross-domain corresponding instances are representative in the source domain and the target domain, respectively. In contrast with previous approaches, in our proposed HHTL framework, we allow the cross-domain instance-correspondences to be biased, and aim to address this issue by using a deep-learning-based architecture such that a precise common feature representation for both the source domain data and the target domain data can still be learned.

5.3. Transfer learning through deep learning

Recently, deep learning techniques [40,41] have been proposed for transfer learning. A common goal of deep learning approaches to transfer learning is to discover high-level features from the original features through a hierarchical structure,

which are supposed to capture the generic factors of variations present in different domains, i.e., the source domain and the target domain.

Raina et al. [42] proposed a self-taught learning framework based on sparse coding [43] to learn high-level features from a huge amount of unlabeled data whose labels can be different from those of the target classification task. Glorot et al. [44] proposed to learn a universal classifier for cross-domain sentiment classification by applying stack denoised autoencoder (SDA) [45] to learn invariant features from large-scale unlabeled data from various product domains. In a follow-up work, Chen et al. [16] proposed a variation of SDA for transfer learning, namely marginalized SDA (mSDA), which has been shown to be more effective and efficient for learning high-level features for transfer learning. Yosinski et al. [46] and Donahue et al. [47] empirically studied how to reuse high-level features extracted from a deep neural network trained on a large-scale dataset or a number of source tasks to learn more powerful high-level features for the target task efficiently and effectively. More recently, Zhuang et al. [48] and Long et al. [49] proposed to encode KL-divergence or Maximum Mean Discrepancy (MMD) [50] into deep autoencoders or deep convolutional neural networks for learning high-level features for transfer learning, respectively. These methods are proposed for homogeneous transfer learning, and thus not applicable for HTL.

Socher et al. [30] proposed a deep-learning-based approach to zero-shot learning, which learns a feature mapping between data of different modalities, i.e., text v.s. image, with a lot of multi-modal data, i.e., pairs of text-image instances, to detect instances of the classes that are not present in training. Though this can be considered as an example of HTL, the correspondences between different modalities need to be fully observed as inputs in training. The focus of our work is learning a precise feature mapping of heterogeneous features between domains for HTL when cross-domain correspondences are insufficient and biased.

6. Conclusions

In this paper, we propose a Hybrid Heterogeneous Transfer Learning (HHTL) framework which allows cross-domain instance-correspondences to be biased to the source domain (and the target domain). Based on the framework, we propose two deep architectures to simultaneously transfer knowledge across different feature spaces through cross-domain feature transformation and correct the data bias issue through high-level features learning. We conduct extensive experiments on a number of cross-language sentiment or document classification tasks to demonstrate the superiority of the proposed HHTL approaches over several baseline methods. In our future work, we plan to extend our framework to apply to other HTL applications, such as text v.s. images applications.

Declaration of Competing Interest

There is no competing interest.

Acknowledgements

This work is partially supported by the NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020, Singapore MOE AcRF Tier-2 grant MOE2016-T2-2-060, and Data Science & Artificial Intelligence Research Centre (DSAIR) at NTU Singapore. Ivor W. Tsang thanks the support of the Australian Research Council Future Fellowship FT130100746 and the Australian Research Council grant LP150100671.

References

[1] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[2] Q. Yang, Y. Chen, G.-R. Xue, W. Dai, Y. Yu, Heterogeneous transfer learning for image clustering via the socialweb, in: ACL/IJCNLP, 2009, pp. 1–9.
[3] S.J. Pan, Transfer learning, in: Data Classification: Algorithms and Applications, 2014, pp. 537–570.
[4] J. Blitzer, R. McDonald, F. Pereira, Domain adaptation with structural correspondence learning, in: EMNLP, 2006, pp. 120–128.
[5] S.J. Pan, J.T. Kwok, Q. Yang, Transfer learning via dimensionality reduction, in: AAAI, 2008, pp. 677–682.
[6] B. Kulis, K. Saenko, T. Darrell, What you saw is not what you get: domain adaptation using asymmetric kernel transforms, in: CVPR, 2011, pp. 1785–1792.
[7] W. Dai, Y. Chen, G.-R. Xue, Q. Yang, Y. Yu, Translated learning: transfer learning across different feature spaces, in: NIPS, 2008, pp. 353–360.
[8] J. Blitzer, M. Dredze, F. Pereira, Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification, in: ACL, 2007, pp. 432–439.
[9] S.J. Pan, X. Ni, J.-T. Sun, Q. Yang, C. Zheng, Cross-domain sentiment classification via spectral feature alignment, in: WWW, 2010, pp. 751–760.
[10] F. Li, S.J. Pan, O. Jin, Q. Yang, X. Zhu, Cross-domain co-extraction of sentiment and topic lexicons, in: ACL, 2012, pp. 410–419.
[11] A. Blum, T.M. Mitchell, Combining labeled and unlabeled data with co-training, in: COLT, 1998, pp. 92–100.
[12] D.R. Hardoon, S. Szedmák, J. Shawe-Taylor, Canonical correlation analysis: an overview with application to learning methods, Neural Comput. 16 (12) (2004) 2639–2664.
[13] M. Amini, N. Usunier, C. Goutte, Learning from multiple partially observed views - an application to multilingual text categorization, in: NIPS, 2009, pp. 28–36.
[14] V. Sindhwani, D.S. Rosenberg, An RKHS for multi-view learning and manifold co-regularization, in: ICML, 2008, pp. 976–983.
[15] J.T. Zhou, S.J. Pan, I.W. Tsang, Y. Yan, Hybrid heterogeneous transfer learning through deep learning, in: AAAI, 2014, pp. 2213–2220.
[16] M. Chen, Z.E. Xu, K.Q. Weinberger, F. Sha, Marginalized denoising autoencoders for domain adaptation, in: ICML, 2012.
[17] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Neurocomputing: Foundations of Research, Ch. Learning Representations by Back-Propagating Errors, MIT Press, Cambridge, MA, USA, 1988, pp. 696–699.

[18] Y.A. LeCun, L. Bottou, G.B. Orr, K.-R. Müller, Efficient backprop, in: Neural Networks: Tricks of the Trade, Springer, 2012, pp. 9–48.
[19] M. Chen, K. Weinberger, F. Sha, Y. Bengio, Marginalized denoising auto-encoders for nonlinear representations, in: ICML, 2014, pp. 1476–1484.
[20] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: a system for large-scale machine learning, in: OSDI, 2016, pp. 265–283.
[21] F. Chollet, et al., Keras, https://keras.io, 2015.
[22] P. Prettenhofer, B. Stein, Cross-language text classification using structural correspondence learning, in: ACL, 2010, pp. 1118–1127.
[23] A. Vinokourov, J. Shawe-Taylor, N. Cristianini, Inferring a semantic representation of text via cross-language correlation analysis, in: NIPS, 2002, pp. 1473–1480.
[24] X. Shi, Q. Liu, W. Fan, P.S. Yu, R. Zhu, Transfer learning on heterogenous feature spaces via spectral transformation, in: ICDM, 2010, pp. 1049–1054.
[25] M. Xiao, Y. Guo, A novel two-step method for cross language representation learning, in: NIPS, 2013, pp. 1259–1267.
[26] C. Wang, S. Mahadevan, Heterogeneous domain adaptation using manifold alignment, in: IJCAI, 2011, pp. 1541–1546.
[27] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: a library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874.
[28] S. Ben-David, J. Blitzer, K. Crammer, F. Pereira, Analysis of representations for domain adaptation, in: NIPS, 2006, pp. 137–144.
[29] J. Heaton, Introduction to Neural Networks with Java, Heaton Research, Inc., 2008.
[30] R. Socher, M. Ganjoo, C.D. Manning, A.Y. Ng, Zero-shot learning through cross-modal transfer, in: NIPS, 2013, pp. 935–943.
[31] S.J. Pan, I.W. Tsang, J.T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, in: IJCAI, 2009, pp. 1187–1192.
[32] S.J. Pan, I.W. Tsang, J.T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE Trans. Neural Netw. 22 (2) (2011) 199–210.
[33] A.J. Smola, A. Gretton, L. Song, B. Schölkopf, A Hilbert space embedding for distributions, in: ALT, 2007, pp. 13–31.
[34] H. Daumé III, Frustratingly easy domain adaptation, in: ACL, 2007, pp. 256–263.
[35] H. Daume III, A. Kumar, A. Saha, Co-regularization based semi-supervised domain adaptation, in: NIPS, 2010, pp. 478–486.
[36] B. Gong, Y. Shi, F. Sha, K. Grauman, Geodesic flow kernel for unsupervised domain adaptation, in: CVPR, 2012, pp. 2066–2073.
[37] L. Duan, D. Xu, I.W. Tsang, Learning with augmented features for heterogeneous domain adaptation, in: ICML, 2012.
[38] J.T. Zhou, I.W. Tsang, S.J. Pan, M. Tan, Heterogeneous domain adaptation for multiple classes, in: AISTATS, 2014, pp. 1095–1103.
[39] W. Pan, Q. Yang, Transfer learning in heterogeneous collaborative filtering domains, Artif. Intell. 197 (2013) 39–55.
[40] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[41] Y. Bengio, A.C. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[42] R. Raina, A. Battle, H. Lee, B. Packer, A.Y. Ng, Self-taught learning: transfer learning from unlabeled data, in: ICML, 2007, pp. 759–766.
[43] H. Lee, A. Battle, R. Raina, A.Y. Ng, Efficient sparse coding algorithms, in: NIPS, 2006, pp. 801–808.
[44] X. Glorot, A. Bordes, Y. Bengio, Domain adaptation for large-scale sentiment classification: a deep learning approach, in: ICML, 2011, pp. 513–520.
[45] P. Vincent, H. Larochelle, Y. Bengio, P. Manzagol, Extracting and composing robust features with denoising autoencoders, in: ICML, 2008, pp. 1096–1103.
[46] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: NIPS, 2014, pp. 3320–3328.
[47] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: a deep convolutional activation feature for generic visual recognition, in: ICML, 2014, pp. 647–655.
[48] F. Zhuang, X. Cheng, P. Luo, S.J. Pan, Q. He, Supervised representation learning: transfer learning with deep autoencoders, in: IJCAI, 2015, pp. 4119–4125.
[49] M. Long, Y. Cao, J. Wang, M. Jordan, Learning transferable features with deep adaptation networks, in: ICML, 2015, pp. 97–105.
[50] A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, J. Mach. Learn. Res. 13 (1) (2012) 723–773.