Neurocomputing 390 (2020) 99–107

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

DT-LET: Deep transfer learning by exploring where to transfer☆

Jianzhe Lin[a], Liang Zhao[b], Qi Wang[c,*], Rabab Ward[a], Z. Jane Wang[a]

[a] Department of Electrical and Computer Engineering, University of British Columbia, Canada
[b] Software Engineering from Dalian University of Technology, China
[c] Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, China

ARTICLE INFO

Article history:
Received 4 November 2019
Revised 31 December 2019
Accepted 10 January 2020
Available online 23 January 2020

Communicated by Dr. Lu Xiaoqiang

Keywords:
Transfer learning
Where to transfer
Deep learning

ABSTRACT

Previous transfer learning methods based on deep network assume the knowledge should be transferred between the same hidden layers of the source domain and the target domains. This assumption doesn't always hold true, especially when the data from the two domains are heterogeneous with different resolutions. In such case, the most suitable numbers of layers for the source domain data and the target domain data would differ. As a result, the high level knowledge from the source domain would be transferred to the wrong layer of target domain. Based on this observation, "where to transfer" proposed in this paper might be a novel research area. We propose a new mathematic model named DT-LET to solve this heterogeneous transfer learning problem. In order to select the best matching of layers to transfer knowledge, we define specific loss function to estimate the corresponding relationship between high-level features of data in the source domain and the target domain. To verify this proposed cross-layer model, experiments for two cross-domain recognition/classification tasks are conducted, and the achieved superior results demonstrate the necessity of layer correspondence searching.

1. Introduction

Transfer learning or domain adaption aims at digging potential information in auxiliary source domain to assist the learning task in target domain, where insufficient labeled data with prior knowledge exist [1]. Especially for tasks like image classification or recognition, the labeled data (the target domain data) are highly required but always not enough as labeling process would be quite tedious and laborious. Without the help of related data (the source domain data), the learning tasks will fail. Therefore, having a better use of auxiliary source domain data by transfer learning methods has attracted researchers' attention.

It should be noted direct use of labeled source domain data on a new scene of target domain would result in poor performance due to the semantic gap between the two domains, even they are representing the same objects [2–5]. The semantic gap can be resulted from different acquisition conditions(illumination or view angle), and the use of different cameras or sensors. Transfer learning methods are proposed to overcome this semantic gap [6–9]. Traditionally, these transfer learning methods would adopt linear or non-linear transformation with kernel function to learn a common subspace on which the gap is bridged [10–12]. Recent advancement has proven that the features learnt on such common subspace are inefficient. Therefore, deep learning based model has been introduced due to its power on high level feature representation.

Current deep learning based transfer learning topics include two research branches, what knowledge to transfer and how to transfer knowledge [13]. For what knowledge to transfer, researchers mainly concentrate on instance-based transfer learning and parameter transfer approaches. Instance-based transfer learning methods assume that only certain parts of the source data can be reused for learning in the target domain by re-weighting [14]. As for parameter transfer approaches, people mainly try to find the pivot parameters in deep network to transfer to accelerate the transfer process. For how to transfer knowledge, different deep networks are introduced to complete the transfer learning process. However, for both research areas, the right correspondence of layers is ignored.

2. Limitations and contributions

The current limitation of transfer learning problems are concluded below. For what knowledge to transfer problem, the transferred content might even be negative or wrong. A fundamental problem for current transfer learning work should be negative

☆ DT-LET can be pronounced as "Data Let", meaning let data to be transferred to the right place, in accord with "Where to Transfer"

* Corresponding author at: Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, China.
E-mail address: crabwq@nwpu.edu.cn (Q. Wang).

https://doi.org/10.1016/j.neucom.2020.01.042
0925-2312/

transfer [15]. If the knowledge from the source domain to target domain is transferred to wrong layers, the transferred knowledge is quite error-prone. With the wrong prior information added, bad effect can be generated on target domain data. For how to transfer knowledge problem, as the two deep networks for the source domain data and the target domain data need to have the same number of layers, the two models could not be optimal at the same time. This situation is especially important for cross-resolution heterogeneous transfer. For data with different resolutions, The data with higher resolution might need more max-pooling layers than the data with lower resolution, and more neural network layers are needed. Based on the above observation and assumption, we propose a new research topic, where to transfer. In this work, the number of layers for two domains does not need to be the same, and optimal matching of layers will be found by the newly proposed objective function. With the best parameters from the source domain data transferred to the right layer of the target domain, the performance of the target domain learning task can be improved.

The proposed work is named Deep Transfer Learning by Exploring where to Transfer (DT-LET), which is based on Stacked Auto-Encoders [16]. A detailed flowchart is shown in Fig. 1. The main contributions are concluded as follows.

- This paper for the first time introduces the where to transfer problem. The deep networks from the source domain and the target domain no longer need to be with the same parameter settings, and the cross-layer transfer learning is proposed in this paper.
- We propose a new principle for finding the correspondence between neural networks in the source domain and in the target domain by defining new unified objctive loss function. By optimizing this objective function, the best setting of two deep networks as well as the correspondence relationship can be figured out.

3. Related work

Deep learning intends to learn nonlinear representation of raw data to reveal the hidden features [1–36]. However, a large number of labeled data are required to avoid over-fitting during the feature learning process. To achieve this goal, transfer learning has been introduced to augment the data with prior knowledge. By aligning data from different domains to high-level correlation space, the data information on different domains can be shared. To find this correlation space, many deep transfer learning frameworks have been proposed in recent years. The main motivation is to bridge the semantic gap between the two deep neural networks of the source domain and the target domain. However, due to the complexity of transfer learning, some transfer mechanisms still lack satisfying interpreting. Based on this consideration, quite a few interesting ideas have been generated. To solve how to determine which domain to be source or target problem, Fabio et al. [19] propose to automatically align domains for the source and target domain. To boost the transfer efficiency and find extra profit during the transfer process, deep mutual learning [20] has been proposed to transfer knowledge bidirectionally. The function of each layer in transfer learning is explored in [21]. The transfer learning with unequal classes and data are experimented in [22] and [23] respectively. However, all the above works still just explain what knowledge to transfer and how to transfer knowledge problems. They still ignore interpreting the matching mechanisms between layers of deep networks of the source domain and the target domain. For this problem, We also name it as DT-LET: Deep Transfer Learning by Exploring Where to Transfer. For this work, we adopt stacked denoising autoencoder(SDA) as the baseline deep network for transfer learning.

Glorot et al. for the first time employ stacked denoising to learn homogeneous features based on joint space for sentiment classification [24]. The computation complexity is further reduced by Chen et al. by the proposing of Marginalized Stacked Denoising Autoencoder(mSDA) [25]. In this work, some characteristics of word vector are set to zero in the equations of expectation to optimize the representation. Still by matching the marginal as well as conditional distribution, Zhang et al. and Zhuang et al. also develop SDA based homogeneous transfer learning framework [26,27]. For heterogeneous case, Zhou et al. [28] propose an extension of mSDA to bridge the semantic gap by finding the cross-domain corresponding instances in advance. Google brain team in recent time introduces generative adversarial network to SDA and propose the Wasserstein Auto-Encoders [29] to generate samples of better quality on target domain. It can be found SDA is with quite high potential, and our work also chooses SDA as the basic neural network for the where to transfer problem.

4. Deep mapping mechanism

The general framework of such deep mapping mechanism can be summarized as three steps, network setting up, correlation maximization, and layer matching. We would like to first introduce the deep mapping mechanism by defining the variables.

The samples in the source domain are denoted as $D_S = \{I_i^s\}_{i=1}^{n_s}$, in which the labeled data in the source domain is further denoted as $D_S^l = \{X_i^s, Y_i^s\}_{i=1}^{n_l}$, they are used to supervise the classification process. In the target domain, the samples are demoted as $D_T = \{I_i^t\}_{i=1}^{n_t}$. The co-occurrence data [30] (the data in the source domain and the target domain belonging to the same classes but with no prior label information) in the source domain are denoted as $C_S = \{C_i^s\}_{i=1}^{n_c}$, in target domain are denoted as $C_T = \{C_i^t\}_{i=1}^{n_c}$. They are further jointly represented by $D^C = \{C_i^S, C_i^T\}_i^{n_c}$, which are used to supervise the transfer learning process. The parameters of deep network in the source domain are denoted by $\Theta^S = \{W^s, b^s\}$, and $\Theta^T = \{W^t, b^t\}$ in the target domain.

The matching of layers is denoted by $R_{s,t} = \{r^1_{i_1,j_1}, r^2_{i_2,j_2}, \ldots, r^m_{i_a,j_b}\}$, in which a represents the total number of layers for the source domain data, and b represents the total layers for the target domain data. m is the total number of matching layers. We define here, if $m = \min\{a-1, b-1\}$(as the first layer is the original layer which will not be used to transfer, m is compared with a-1 or b-1 instead of a or b), we define the transfer process as full rank transfer learning; else if $m < \min\{a-1, b-1\}$, we define this case as non-full rank transfer learning.

The common subspace is represented by Ω and the final classifier is represented by Ψ. The labeled data D_S^l from the source domain are used to predict the label of D_T by applying $\Psi(\Omega(D_T))$.

4.1. Network setting Up

The stacked auto-encoder (SAE) is first employed in the source domain and the target domain to get the hidden feature representation H^S and H^T of original data as shown in Eqs. (1) and (2).

$$\begin{aligned} &H^S(n+1) = f(W^S(n) \times H^S(n) + b^S(n)), n > 1; \\ &H^S(n) = f(W^S(n) \times C_S + b^S(n)), n = 1. \end{aligned} \tag{1}$$

$$\begin{aligned} &H^T(n+1) = f(W^T(n) \times H^T(n) + b^T(n)), n > 1; \\ &H^T(n) = f(W^T(n) \times C_T + b^T(n)), n = 1. \end{aligned} \tag{2}$$

Here W^S and b^S are parameters from neural network Θ^S, W^T and b^T are parameters from neural network Θ^T. $H^S(n)$ and $H^T(n)$

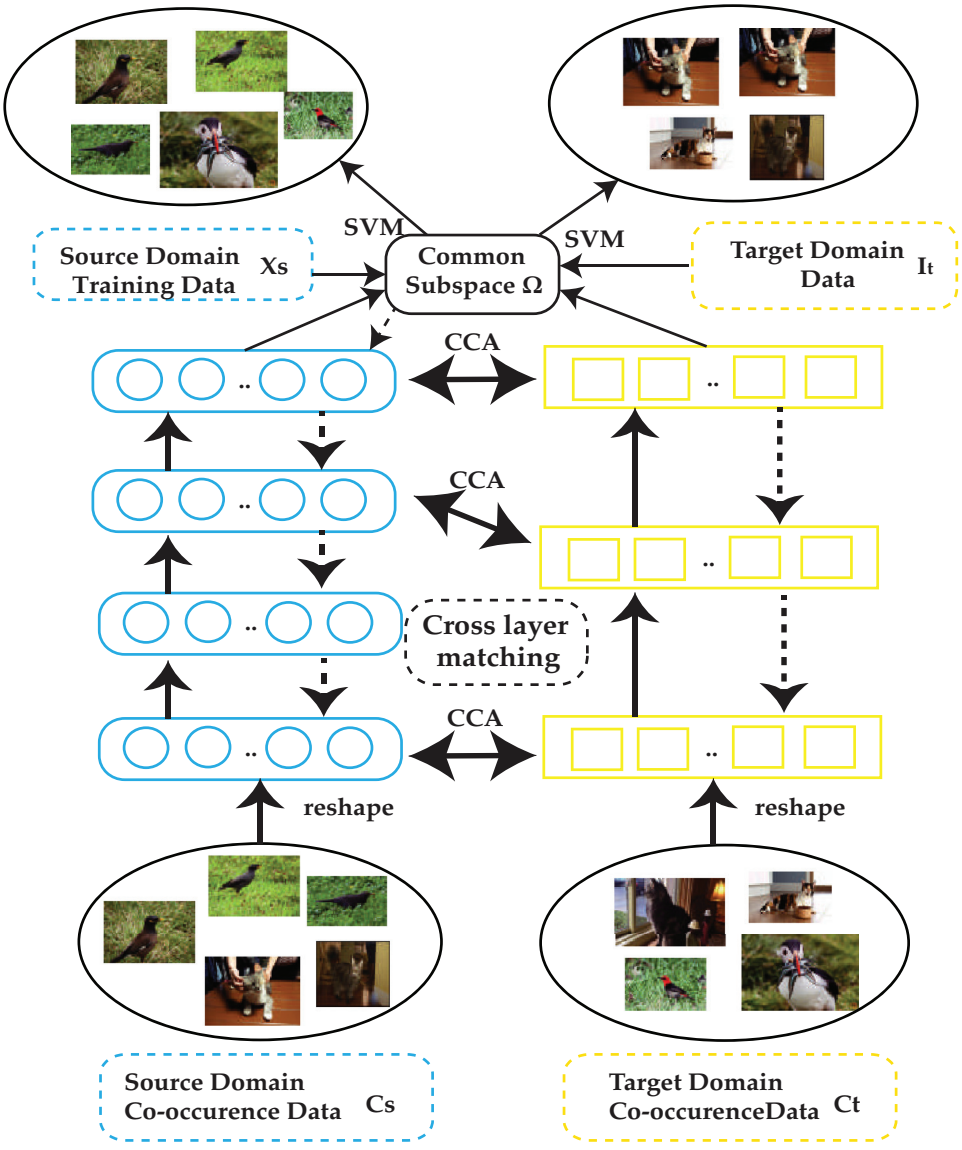


Fig. 1. The flowchart of the proposed DT-LET framework. The two neural networks are first trained by the co-occurrence data C_s and C_t. After network training, the common subspace is found and the training data D_S^l is transferred to such space to train SVM classifier, to classify D_T.

mean the nth hidden layers in the source domain and in the target domain respectively. The two neural networks are first initialized by above functions.

4.2. Correlation maximization

To set up the initial relationship of the two neural networks, we resort to Canonical Correlation Analysis (CCA) which can maximize the correlation between two domains [31]. A multi-layer correlation model based on the above deep networks is further constructed. Both the C_S and the C_T are projected by CCA to a common subspace Ω on which a uniformed representation is generated. Such projection matrices obtained by CCA are denoted as $V^S(n)$ and $V^T(n)$. To find optimal neural networks in the source domain and in the target domain, we have two general objectives: to minimize the reconstruction error of neural networks of the source domain and the target domain, and to maximize the correlation between the two neural networks. To achieve the second objective, we need on one hand find the best layer matching, on the other hand maximize the correlation between corresponding layers. To achieve this goal, we can minimize the final objective function

$$L(R_{s,t}) = \frac{L_s(\theta^S) + L_T(\theta^T)}{P(V^S, V^T)}, \tag{3}$$

in this function, the objective function is defined as L, and $L(R_{s,t})$ is in corresponding with different matching of $R_{s,t}$. We would like to generate the best matching by finding the minimum $L(R_{s,t})$. In $L(R_{s,t})$, $L_s(\Theta^S)$ and $L_T(\Theta^T)$ represent the reconstruction errors of data in the source domain and the target domain, which are defined as follows:

$$L_S(\theta^S) = \left[\frac{1}{n_s}\sum_{i=1}^{n_s}\left(\frac{1}{2}||h_{W^S,b^S}(C_i^s) - X_i^s||^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n^S-1}\sum_{j=1}^{n_l^S}\sum_{k=1}^{n_{l+1}^S}(W_{kj}^{S(l)})^2 \tag{4}$$

$$L_T(\theta^T) = \left[\frac{1}{n_t}\sum_{i=1}^{m_t}\left(\frac{1}{2}||h_{W^T,b^T}(C_i^t) - C_i^t||^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n^T-1}\sum_{j=1}^{n_l^T}\sum_{k=1}^{n_{l+1}^T}(W_{kj}^{T(l)})^2, \tag{5}$$

Here n^S and n^T are the numbers of their layers, n_l^S and n_l^S are the numbers of neurons in layer l, and λ is the trade-off parameter. The third term $P(V_s, V_t)$ represents the domain divergence after projection by CCA which we want to maximize. The definition for this term is in Eq. (6)

$$P(V^S, V^T) = \sum_{l=2}^{n^S-1} \frac{V^{S(l)^T} \sum_{ST} V^{T(l)}}{\sqrt{V^{S(l)^T} \sum_{SS} V^{S(l)}} \sqrt{V^{T(l)^T} \sum_{TT} V^{T(l)}}}, \tag{6}$$

where $\sum_{ST} = H^{S(l)} H^{T(l)^T}$, $\sum_{SS} = H^{S(l)} H^{S(l)^T}$, $\sum_{TT} = H^{T(l)} H^{T(l)^T}$. By minimizing Eq. (3), we can collectively train the two neural networks $\theta^T = \{W^T, b^T\}$ and $\theta^S = \{W^S, b^S\}$.

4.3. Layer matching

After constructing the multiple layers of the networks by Eq. (3), we need to further find the best matching for layers after construction of neural networks. As different layer matching would generate different function loss value L in Eq. (3), we define the objective function for layer matching as

$$R_{s,t} = \arg\min L. \tag{7}$$

5. Model training

Here we would like to first optimize the Eq. (3). As the equation is not joint convex with all the parameters θ^S, θ^T, V_s, and V_t, and the two parameters θ^S and θ^T are not related with V^S and v^T, we would like to introduce two-step iteration optimization.

5.1. Step.1: Updating V^S, V^T with fixed Θ^S, Θ^T

In Eq. (3), the optimization of V^S, V^T is just related to the dominator term. The optimization of each layer $V^S(l_1)$, $V^T(l_2)$(suppose the layer 1_1 on source domain is in corresponding with layer l_2 on target domain) can be formulated as

$$\max_{V^{S(l_1)}, V^{T(l_2)}} \frac{V^{S(l_1)^T} \sum_{ST} V^{T(l_2)}}{\sqrt{V^{S(l_1)^T} \sum_{SS} V^{S(l_1)}} \sqrt{V^{T(l_2)^T} \sum_{TT} V^{T(l_2)}}} \tag{8}$$

As $V^{S(l_1)^T} \sum_{SS} V^{S(l_1)} = 1$ and $V^{T(l_2)^T} \sum_{TT} V^{T(l_2)} = 1$ [31], we can rewrite Eq. (8) as

$$\begin{aligned} &\max V^{S(l_1)^T} \sum_{ST} V^{T(l_2)}, \\ &s.t. V^{S(l_1)^T} \sum_{SS} V^{S(l_1)} = 1, V^{T(l_2)^T} \sum_{TT} V^{T(l_2)} = 1 \end{aligned} \tag{9}$$

This is a typical constrained problem which can be solved as a series of unconstrained minimization problems. we introduce the Lagrangian multiplier to solve this problem and we have

$$\begin{aligned} L(w_l, V^{S(l_1)}, V^{T(l_2)}) &= V^{S(l_1)^T} \sum_{ST} V^{T(l_2)} \\ &+ \frac{w_l^S}{2} \left(V^{S(l_1)^T} \sum_{SS} V^{S(l_1)} - 1 \right) \\ &+ \frac{w_l^T}{2} \left(V^{T(l_2)^T} \sum_{TT} V^{T(l_2)} - 1 \right) \end{aligned} \tag{10}$$

Then we take the partial derivatives for V^S Eq. (10) and get

$$\frac{\partial L}{\partial V^{S(l_1)}} = \sum_{ST} V^{T(l_2)} - w_l^S \sum_{SS} V^{S(l_1)} = 0. \tag{11}$$

The partial derivatives for V^T is the same. Now we can have the final solution as

$$\sum_{ST} \sum_{TT}^{-1} \sum_{ST}^{T} V^{S(l_1)} = w_l^2 \sum_{SS} V^{S(l_1)}, \tag{12}$$

here we assume $w_l = w_l^S = w_l^T$. From here $V^S(l_1)$ and w_l can be solved by the generalized eigenvalue decomposition and we can also get the corresponding $V^T(l_2)$.

5.2. Step.2: Updating Θ^S, Θ^T with fixed V^S, V^T

As Θ^S and Θ^T are mutual independent and with the same form, we here just demonstrate the solution of Θ^S on the source domain (the solution of Θ^T can be derived similarly). Actually the objective division operation is with the same function with subtraction operation and we reformulate the objective function as

$$\min_{\theta^S} \phi(\theta^S) = L_S(\theta^S) - \Gamma(V^S, V^T) \tag{13}$$

Here we apply the gradient descent method to adjust the parameter as

$$\begin{aligned} W^{S(l_1)} &= W^{S(l_1)} - \mu^S \frac{\partial \phi}{\partial W^{S(l_1)}} \\ &= \frac{\partial L_S(\theta^S)}{\partial W^{S(l_1)}} - \frac{\partial \Gamma(V^S, V^T)}{\partial W^{S(l_1)}} \\ &= \frac{(\alpha^{S(l_1+1)} - \beta^{S(l_1+1)} + \omega_l \gamma^{S(l_1+1)}) \times H^{S(l_1)}}{n_c + \lambda^S W^{S(l_1)}} \end{aligned} \tag{14}$$

$$\begin{aligned} b^{S(l_1)} &= b^{S(l_1)} - \mu^S \frac{\partial \phi}{\partial b^{S(l_1)}} \\ &= \frac{\partial L_S(\theta^S)}{\partial b^{S(l_1)}} - \frac{\partial \Gamma(V^S, V^T)}{\partial b^{S(l_1)}} \\ &= \frac{(\alpha^{S(l_1+1)} - \beta^{S(l_1+1)} + \omega_l \gamma^{S(l_1+1)})}{n_c}, \end{aligned} \tag{15}$$

in which

$$\alpha^{S(l_1)} = \begin{cases} -(D_S^l - H^{S(l_1)}) \cdot H^{S(l_1)} \cdot (1 - H^{S(l_1)}), l = n^S \\ W^{S(l_1)^T} \alpha^{S(l_1+1)} \cdot H^{S(l_1)} \cdot (1 - H^{S(l_1)}), \\ l = 2, \ldots, n^S - 1 \end{cases} \tag{16}$$

$$\beta^{S(l_1)} = \begin{cases} 0, \quad l = n^S \\ H^{T(l_2)} V^{T(l_2)} V^{S(l_1)^T} \cdot H^{S(l_1)} \cdot (1 - H^{S(l_1)}), \\ l = 2, \ldots, n^S - 1 \end{cases} \tag{17}$$

$$\gamma^{S(l_1)} = \begin{cases} 0, l = n^S \\ H^{S(l_1)} V^{S(l_1)} V^{S(l_1)^T} \cdot H^{S(l_1)} \cdot (1 - H^{S(l_1)}), . \\ l = 2, \ldots, n^S - 1 \end{cases} \tag{18}$$

The operator $\cdot$ here stands for the dot product. The same optimization process works for Θ^T on the target domain.

After these two optimizations for each layer, the two whole networks (the source domain network and the target domain network) are further fine-tuned by the back-propagation process. The forward and backward propagations will iterate until convergence.

5.3. Optimization of $R_{s,t}$

We finally get the minimized $L(R_{s,t})$ by the above procedure. As described previously, we have $a-1$ layers used for source domain and $b-1$ layers for target domain, the expected computation complexity of exhaustive search can be approximated by $O(a \times b)$, and the problem should be NP-hard to optimize. To reduce this to polynomial complexity, we introduce the Immune Clonal Strategy(ICS) to solve this problem. We take Eq. (7) as the affinity function in the ICS. The source domain layers are regarded as the antibodies, and the target domain layers are viewed as the antigen. Various antibodies have different effectiveness for the antigen. By maximizing the affinity function, the best antibodies are chosen. The detailed optimization procedure is modeled as an iterative process. It includes three phases: clone, mutation, and selection.

5.3.1. Clone phase

At the very first, we set the $a-1$ layers on the source domain as the antibodies $E(m)=\{e_1(m),e_2(m),\ldots,e_{a-1}(m)\}$. For random $e_i(t)$ we have b(as $e_i(t)=0$ is also one choice) choices of values, representing the corresponding layers in the target domain. For notational simplicity, we omit the iteration number m in the following explanation with no loss of understandability. The initial $E(m)$ is cloned for n_c times and we get $\mathscr{E}=\{E^1,E^2,\ldots,E^{n_c}\}$.

5.3.2. Mutation phase

The randomly chosen antibodies are not the best. Therefore, a mutation phase after the clone phase is necessary. For example, for the antibody E^i, we randomly replace N_m representatives in its clone E^i by the same number of elements. There is no doubt that these newly introduced elements should differ from the former representatives, which enrich the diversity of the original antibodies. After this, mutated antibodies $\mathscr{E}=\{E^1,E^2,\ldots,E^{n_c}\}$ are obtained.

5.3.3. Selection phase

With the obtained antibodies, which are manifestly more various than the original set, we will select the most promising ones for the next round of processing. The principle is also defined with the affinity values. Higher ones indicate more fitness. Therefore,we have

$$\mathrm{E}(m+1)=argmax\{S(E_1),S(E_2),\ldots,S(E_{n_c})\} \tag{19}$$

which means that the antibody with the largest affinity value is taken as $E(m+1)$ to enter the next iteration. The iteration does not terminate until the change between $S(E(m))$ and $S(E(m+1))$ is smaller than threshold or the maximum number of iteration m_{it} is reached.

The final $E(m)$ then is output as the minimized $R_{s,t}$. However, after our experiments, we heuristically find the number of matching layers would almost be in direct proportion to the resolution of images.

The training process is finally summarized in Algorithm (1).

5.4. Classification on common semantic subspace

The final classification is performed on the common subspace Ω. The unlabeled data on the target domain D^T and the labeled D^S are both projected to the common subspace Ω by the correlation coefficients $V^S(n_S)$ and $V^T(n_T)$. The projection is formulated as $H^S=A^S(n_S)V^S(n_S)$ in the source domain, and $H^T=A^T(n_T)V^T(n_T)$ in the target domain. The standard SVM algorithm is applied on Ω. The classifier Ψ is trained by $\{H_i^S,Y_i^S\}_i^{n_S}$. This trained classifier Ψ is applied to D^T as $\Psi(H^T)$. The pseudo-code of the proposed approach can be found in Algorithm (2) below.

Algorithm 1 Deep Mapping Model Training.

Input: $D^C=\{C_i^s,C_i^t\}_i^{n_c}$,
Input: $\lambda^S=1,\lambda^T=1,\ \mu^S=0.5,\mu^T=0.5$
Output: $\Theta(W^S,b^S),\Theta(W^T,b^T),V^S,V^T$

function NetworkSetUp
  Initialize $\Theta(W^S,b^S),\Theta(W^T,b^T)\leftarrow RandomNum$
  repeat
    for $l=1,2,\ldots,n^S$ **do**
      $V^S\leftarrow\arg\min L(\omega_l,V^{S(l)})$
    end for
    for $l=1,2,\ldots,n^T$ **do**
      $V^T\leftarrow\arg\min L(\omega_l,V^{T(l)})$
    end for
    $\theta^S=\arg\min\phi(\theta^S),\theta^T=\arg\min\phi(\theta^T)$
  until Convergence
end function
function LayerMatching
  Initialize $R_{s,t}\leftarrow RandomMatching$
  Initialize $m\leftarrow 0,\sigma\leftarrow 1$
  if $m<m_{it}$ **then**
    if $\sigma>0.05$ **then**
      $R_{s,t}^m=E(m)=\{e_1(t),e_2(t),\ldots,e_{a-1}(t)\}$
      $\mathscr{E}=\{E^1,E^2,\ldots,E^{n_c}\}$
      $\mathrm{E}(m+1)=argmax\{S(E_1),S(E_2),\ldots,S(E_{n_c})\}$
      m = m+1
    end if
  end if
end function

Algorithm 2 Classification on Common Semantic Subspace.

Input: X^S,Y^S,V^S,X^T,V^T
Input: $\Theta(W^S,b^S),\Theta(W^T,b^T),n^s,n^t$
Output: Y^T

function SVMTRAINING($X^S,Y^S,V^S,\Theta(W^S,b^S),n^s$)
  for $i=1,2,3,..,n^s$ **do**
    Calculate $H^S(n^s)$ for $X^S(n^s)$ by $\Theta(W^S,b^S)$ as Eq. (1)
    $A^S\leftarrow H^S(n^s)V^S(n^s)$
  end for
  $\Psi\leftarrow\{A^S,Y^S\}$
end function
function SVMTESTING($X^T,V^T,\Theta(W^T,b^T),n^t$)
  for $j=1,2,3,..,n^T$ **do**
    Calculate $H^T(n^t)$ for $X^T(n^t)$ by $\Theta(W^T,b^T)$ as Eq. (2)
    $A^T\leftarrow H^T(n^t)V^T(n^t)$
    $Y^T\leftarrow\Psi(A^T)$
  end for
end function

6. Experiments

We carry out our DT-LET framework on two cross-domain recognition tasks, handwritten digit recognition, and text-to-image classification.

6.1. Experimental dataset descriptions

Handwritten digit recognition: For this task, we mainly conduct the experiment on Multi Features Dataset collected from UCI machine learning repository. This dataset consists of features of handwritten numerals (0–9, in total 10 classes) extracted from a collection of Dutch utility maps. 6 features exist for each numeral and we choose the most popular features 216-D profile correlations and 240-D pixel averages in 2*3 windows to complete the transfer learning based recognition task.

Text-to-image classification: For this task, we make use of NUS-WIDE dataset. In our experiment, the images in this dataset are represented with 500-D visual features and annotated with 1000-D text tags from Flickr. 10 categories of instances are included in this classification task, which are birds, building, cars, cat, dog, fish, flowers, horses, mountain, and plane.

6.2. Comparative methods and evaluation

As the proposed ET-LET framework mainly have four components, deep learning, CCA, layer matching, and SVM classifier, we first select the baseline method, Deep-CCA-SVM(DCCA-SVM) [25] as baseline comparison methods. We also conduct experiment just without layer matching(the number of layers are the same on the source and the target domains) while all the other parameters

Table 1
Classification accuracy results on multi feature dataset. (The best performance is emphasized by boldface).

numeral	DCCA-SVM	duft-tDTNs	DeepCoral	DANN	ADDA	NoneDT-LET	DT-LET($r^3_{5,4}$)	DT-LET($r^2_{4,3}$)
0	0.961	0.972	0.864	0.923	0.966	0.983	**0.989**	0.984
1	0.943	0.956	0.805	0.941	0.978	0.964	0.976	**0.982**
2	0.955	0.972	0.855	0.911	0.982	0.979	0.980	**0.989**
3	0.945	0.956	0.873	0.961	0.973	0.966	**0.976**	0.975
4	0.956	0.969	0.881	0.933	0.980	0.980	**0.987**	0.983
5	0.938	0.949	0.815	0.946	0.970	0.958	0.971	**0.977**
6	0.958	0.966	0.893	0.968	0.961	0.978	**0.988**	0.986
7	0.962	0.968	0.847	0.929	0.979	0.978	0.975	**0.985**
8	0.948	0.954	0.904	0.968	0.971	0.965	0.968	**0.975**
9	0.944	0.958	0.915	0.963	0.969	0.970	**0.976**	0.961

Table 2
Classification Accuracy Results on NUS-WIDE Dataset. (The best performance is emphasized by boldface (method names abbreviated)).

categories	DC.	DC	DA.	d-tD	AD.	N-T	DT-LET			
							$r^2_{5,3}(1)$	$r^2_{5,3}(2)$	$r^2_{5,4}$	$r^3_{5,4}$
birds	0.78	0.77	0.67	0.81	0.81	0.78	0.83	0.83	**0.85**	0.83
building	0.81	0.78	0.67	0.83	0.83	0.82	0.88	0.84	0.88	**0.89**
cars	0.80	0.77	0.69	0.81	0.85	0.81	0.83	0.83	**0.87**	0.85
cat	0.80	0.77	0.78	0.83	0.81	0.81	0.87	**0.87**	0.86	**0.87**
dog	0.80	0.77	0.70	0.82	0.81	0.81	0.85	0.85	**0.86**	0.82
fish	0.77	0.75	0.73	0.82	0.79	0.78	0.85	0.84	**0.85**	0.84
flowers	0.80	0.78	0.77	0.84	0.81	0.81	0.86	0.84	0.84	**0.88**
horses	0.80	0.78	0.72	0.82	0.83	0.81	**0.84**	0.81	**0.84**	0.83
mountain	0.82	0.79	0.75	0.82	0.82	0.83	0.83	0.81	0.82	**0.83**
plane	0.82	0.79	0.79	0.82	0.79	0.83	0.81	**0.83**	**0.83**	0.83
average	0.80	0.77	0.77	0.83	0.83	0.81	0.84	0.83	**0.85**	0.85

are the same with the proposed ET-LET, and we name this framework NoneDT-LET.

The other deep learning based comparison methods are duft-tDTNs [32], DeepCoral [33], DANN [34], and ADDA [35] methods. For these methods, duft-tDTNs is the most representative, which should be up to now heterogenous transfer learning method with best performance. DeepCoral should be the first deep learning based domain adaptation framework, DANN should for the first time introduce the domain adversarial concept to domain adaptation, and ADDA is the most famous unsupervised domain adaptation method.

For the deep network based method, the DCCA-SVM, duft-tDTNs, DeepCoral, DANN, NoneDT-LET are all with 4 layers for the source domain and the target domain data, as we find more or less layer would generate worse performance.

At last, for the evaluation metric, we select the classification accuracies on the target domain data over the 2 pairs of datasets.

6.3. Task 1: Handwritten digit recognition

In the first experiment, we conduct our study for handwritten digit recognition. The source domain data are the 240-D pixel averages in 2*3 windows feature, while the target domain data are the 216-D profile correlations feature. As there are 10 classes in total, we complete 45 (C^2_{10}) binary classification tasks, for each category, the accuracy is the average accuracy of 9 binary classification tasks. We use 60% data as co-occurrence data to complete the transfer learning process and find the common subspace, 20% labeled samples on source domain as the training samples, and the rest samples on target domain as the testing samples to complete the classification process. The experiments are repeated for 100 times with 100 sets of randomly chosen training and testing data to avoid data bias [36]. The final accuracy is the average accuracy of the 100 repeated experiments. This data setting applies for all four methods under comparison.

For the deep network, the numbers of neurons of 4 layer networks are 240-170-100-30 for source domain data and 216-154-92-30 for target domain data, this setting works for the all comparison methods. For the proposed DT-LET, we find the best two layer matching with lowest loss after 25 iterations are $r^2_{4,3}$ and $r^3_{5,4}$. The numbers of neurons for $r^2_{4,3}$ are 240-170-100-30 for source domain data and 216-123-30 for target domain data. The average objective function loss of the all 45 binary classification tasks for these two are 0.856 and 0.832 respectively. The numbers of neurons for $r^3_{5,4}$ are 240-185-130-75-30 for source domain data and 216-154-92-30 for target domain data. The one-against-one SVM classification is applied for final classification. The average classification accuracies of 10 categories are shown in Table 1. The matching correlation is detailed in Fig. 2.

As can be found in Table 1, the best performances have been highlighted, which all exist in DT-LET framework. However, the best performances for different categories do not exist in the framework with same layer matching. Overall, $r^3_{5,4}$ and $r^2_{4,3}$ should be the best two layer matchings compared with other settings. Based on these results, we heuristically get the conclusion that the best layer matching ratio(5/4, 4/3) is generally in direct proportion to the dimension ratio of original data(240/216). However, more matched layers do not guarantee better performance as the classification results for number “1”, “2”, “5”, “7”, “8” of DT-LET ($r^2_{4,3}$) with 2 layer matchings perform better than DT-LET($r^3_{5,4}$) with 3 layer matchings.

6.4. Task 2: Text-to-image classification

In the second experiment, we conduct our study for Text-to-image classification. The source domain data are the 1000-D text feature, while the target domain data are the 500-D image feature. As there are 10 classes in total, we complete 45 (C^2_{10}) binary classification tasks. We still use 60% data as co-occurrence data [30], 20% labeled samples on source domain as the training samples, and the rest samples on target domain as the testing samples.

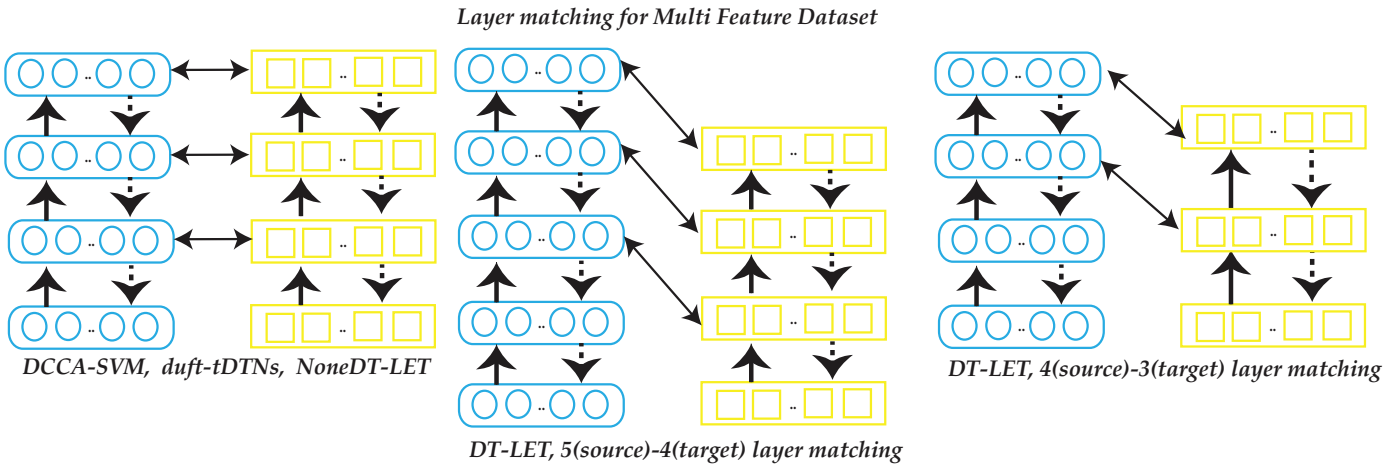


Fig. 2. The comparison of different layer matching setting for different frameworks on Multi Feature Dataset.

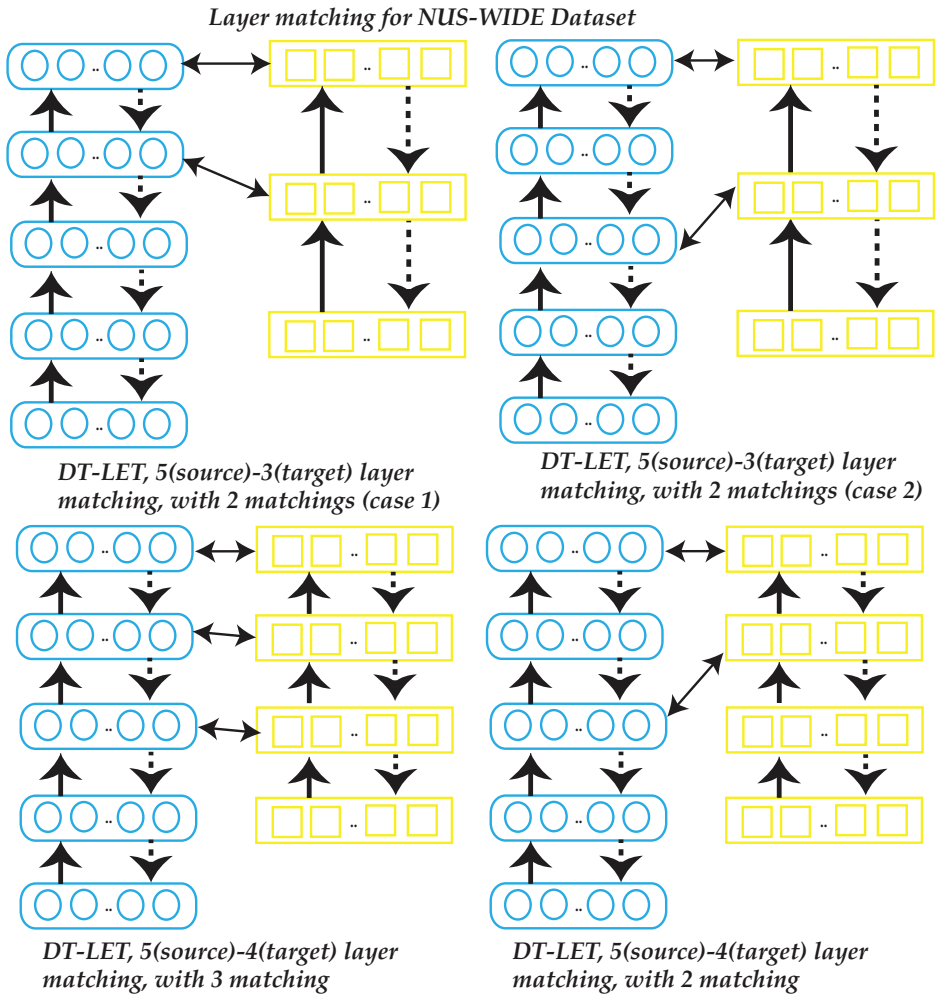


Fig. 3. The comparison of different layer matching setting for different frameworks on NUS-WIDE Dataset.

The same data setting as Task 1 applies for all four methods under comparison.

For the deep network, the numbers of neurons of 4 layer networks are 1000-750-500-200 for source domain data and 500-400-300-200 for target domain data, this setting works for the all comparison methods. For the proposed DT-LET, we find the best two layer matchings with lowest loss after 25 iterations are $r^2_{5,3}$, $r^3_{5,4}$ and $r^2_{5,4}$(non-full rank). The average objective function loss of 45 binary classification tasks for these two layer matchings are 3.231, 3.443 and 3.368. The numbers of neurons for $r^2_{5,3}$ are 1000-800-600–400-200 for source domain data and 500-350-200 for target domain data. The numbers of neurons for both $r^3_{5,4}$ and $r^2_{5,4}$ are 1000-750-500-200 for source domain data and 500-400-300-200 for target domain data. As matching principle would also

influence the performance of transfer learning, we present two $r^2_{5,3}$ with different matching principles as shown in Fig. 3(the average objective function loss for the two different matching principles are 3.231 and 3.455), in which all the detailed layer matching principles are described.

For this task, as the overall accuracies are generally lower than task 1, we would like to compare more different settings for this cross-layers matching task. We first verify the effectiveness of DT-LET framework. Compared with the comparison methods, the accuracy of DT-lET framework is generally with around 85% accuracy while the comparison methods are generally with no more than 80%. This observation generates the conclusion that finding the appropriate layer matching is essential. The second comparison is between the full rank and non-full rank framework. As can be found in the table, actually $r^2_{5,4}$ is with the highest overall accuracy, although the other non-full rank DT-LETs do not perform quite well. This observation gives us a hint that full rank transfer is not always best as negative transfer would degrade the performance. However, the full rank transfer is generally good, although not optimal. The third comparison is between the same transfers with different matching principles. We present two $r^2_{5,3}$ with different matching principles, and we find the performances vary. The case 1 performs better than case 2. This result tell us continuous transfer might be better than discrete transfer: as for case 1, the transfer is in the last two layers of both domains, and in case 2, the transfer is conducted in layer 3 and layer 5 of the source domain data.

By comparing specific objects, we can find the objects with large semantic difference with other categories are with higher accuracy. For the objects which are hard to classify and with low accuracy, like “birds” and “plane”, the accuracies are always low even the DT-LET is introduced. This observation proves the conclusion that DT-LET can only be used to improve the transfer process, which helps with the following classification process; while the classification accuracy is still based on the semantic difference of data of different 10categories.

We also have to point out the relationship between the average objective function loss and the classification accuracy is not strictly positive correlated. Overall $r^2_{5,4}$ is with the highest classification accuracy while its average objective function loss is not lowest. Based on this observation, we have to point out, the lowest average objective function loss can only generate the best transfer leaning result with optimal common subspace. On such common subspace, the data projected from target domain are classified. These classification results are also influenced by the classifier as well as training samples projected randomly from the source domain. Therefore, we conclude as follows. We can just guarantee a good classification performance after getting the optimal transfer learning result, while the classification accuracy is also influenced by the classification settings.

6.5. Parameter sensitivity

In this section, we study the effect of different parameters in our networks. We have to point out the even the layer matching is random, the last layer of the two neural networks from the source domain and the target domain must be correlated to construct the common subspace. Actually, the number of neurons at last layer would also affect the final classification result. For the last layer, we take experiments on Multi Feature Dataset as an example. The result is shown in Table 3.

From this figure, it can be noted when the number of neuron is 30, the performance is the best. Therefore in our former experiments, 30 neurons are used. The conclusion can also be drawn that more neurons are not always better. Based on this observation, The number of layers in Task 1 is set as 30, and in Task 2 as 200.

Table 3
Effects of the number of neurons at the last layer.

layer matching	10	20	30	40	50
$r^3_{5,4}$	0.9082	0.9543	0.9786	0.9771	0.9653
$r^2_{4,3}$	0.8853	0.9677	0.9797	0.9713	0.9522

7. Conclusion

In this paper, we propose a novel framework, referred to as Deep Transfer Learning by Exploring where to Transfer(DT-LET), for hand writing digit recognition and text-to-image classification. In the proposed model, we first find the best matching of deep layers for transfer between the source and target domains. After the matching, the final correlated common subspace is found on which classifier is applied. Experimental results support the effectiveness of the proposed framework.

For the future work, We would propose more robust model to solve the proposed “where to transfer” problem.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Jianzhe Lin: Writing - original draft, Conceptualization, Methodology, Software. **Liang Zhao:** Methodology. **Qi Wang:** Writing - review & editing. **Rabab Ward:** Writing - review & editing. **Z. Jane Wang:** Writing - review & editing.

Acknowlegements

This work was supported by the National Natural Science Foundation of China under Grant U1864204 and 61773316, and Project of Special Zone for National Defense Science and Technology Innovation. This work was also partially supported by NSERC.

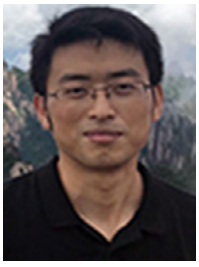
References

[1] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[2] Z. Y., Y. Chen, Z. Lu, S. J. Pan, G. R. Xue, Y. Yu, Q. Yang, Heterogeneous transfer learning for image classification, Proceedings of the AAAI (2011).
[3] L. Duan, D. Xu, W. Tsang, Learning with augmented features for heterogeneous domain adaptation, Proceedings of the ICML (2012).
[4] X. Lu, T. Gong, X. Zheng, Multisource compensation network for remote sensing cross-domain scene classification, IEEE Trans. Geosci. Remote Sens. (2019).
[5] Q. Wang, J. Gao, X. Li, Weakly supervised adversarial domain adaptation for semantic segmentation in urban scenes, IEEE Trans. Image Process. (2019).
[6] W. Dai, Y. Chen, G. Xue, Q. Yang, Y. Yu, Translated learning: Transfer learning across different feature spaces, Proceedings of the NIPS (2009).
[7] T. Liu, Q. Yang, D. Tao, Understanding how feature structure transfers in transfer learning, Proceedings of the IJCAI (2017).
[8] J. Wang, Y. Chen, S. Hao, W. Feng, Z. Shen, Balanced distribution adaptation for transfer learning, Proceedings of the ICDM (2017).
[9] Q. Wang, J. Wan, X. Li, Robust hierarchical deep learning for vehicular management, IEEE Trans. Veh. Technol. 68 (5) (2018) 4148–4156.
[10] Y. Yan, W. Li, M. Ng, M. Tan, H. Wu, H. Min, Q. Wu, Translated learning: Transfer learning across different feature spaces, Proceedings of the IJCAI (2017).
[11] J.S. Smith, B.T. Nebgen, R. Zubatyuk, N. Lubbers, C. Devereux, K. Barros, S. Tretiak, O. Isayev, A.E. Roitberg, Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning, Nature Commun. 10 (1) (2019) 1–8.
[12] Q. Wang, Q. Li, X. Li, Hyperspectral band selection via adaptive subspace partition strategy, IEEE J. Select. Top. Appl. Earth Observat. Remote Sens. (2019).
[13] J. Li, H. Zhang, Y. Huang, L. Zhang, Visual domain adaptation: a survey of recent advances, IEEE Signal Process. Mag. 33 (3) (2015) 53–69.
[14] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, B. Scholkopf, Domain adaptation with conditional transferable components, Proceedings of the ICML (2016).
[15] B. Tan, Y. Zhang, S.J. Pan, Q. Yang, Distant domain transfer learning, Proceedings of the AAAI (2017).

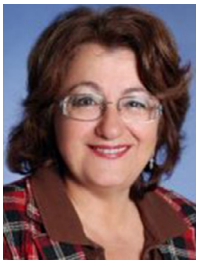
[16] F. Zhuang, X. Cheng, P. Luo, P. J., Q. He, Supervised representation learning with double encoding-layer autoencoder for transfer learning, ACM Trans. Intell. Syst. Technol. (2018) 1–16.
[17] M. Long, J. Wang, Y. Cao, J. Sun, P. Yu, Deep learning of transferable representation for scalable domain adaptation, IEEE Trans. Knowl. Data Eng. 28 (8) (2016) 2027–2040.
[18] Y. Yuan, X. Zheng, X. Lu, Hyperspectral image superresolution by transfer learning, IEEE J. Selected Top. Appl. Earth Observat. Remote Sens. 10 (5) (2017) 1963–1974.
[19] F.M. Carlucci, L. Porzi, B. Caput, Autodial: Automatic domain alignment layers, arXiv:1704.08082 (2017).
[20] Y. Zhang, T. Xiang, T.M. Hospedales, H. Lu, Deep mutual learning, Proceedings of the CVPR (2018).
[21] E. Collier, R. DiBiano, S. Mukhopadhyay, Cactusnets: Layer applicability as a metric for transfer learning, arXiv:1711.01558 (2018).
[22] I. Redko, N. Courty, R. Flamary, D. Tuia, Optimal transport for multi-source domain adaptation under target shift, arXiv:1803.04899 (2018).
[23] M. Bernico, Y. Li, Z. Dingchao, Investigating the impact of data volume and domain similarity on transfer learning applications, Proceedings of the CVPR (2018).
[24] X. Glorot, A. Bordes, Y. Bengio, Domain adaptation for large-scale sentiment classification: A deep learning approach, Proceedings of the ICML (2011).
[25] Y. Yu, S. Tang, K. Aizawa, A. Aizawa, Category-based deep CCA for fine-grained venue discovery from multimodal data, IEEE Trans. Neural Netw. Learn. Syst. (2018) 1–9.
[26] F. Zhuang, X. Cheng, P. Luo, P.S. J., H. Q., Supervised representation learning: Transfer learning with deep autoencoders, Proceedings of the IJCAI (2015).
[27] X. Zhang, F.X. Yu, S.F. Chang, S. Wang, Supervised representation learning: Transfer learning with deep autoencoders, Comput. Sci. (2015).
[28] S.J. Zhou, J. T.and Pan, I.W. Tsang, Y. Yan, Hybrid heterogeneous transfer learning through deep learning, Proceedings of the AAAI (2014).
[29] I. Tolstikhin, O. Bousquet, S. Gelly, B. Schoelkopf, Wasserstein auto-encoders, arXiv:1711.01558, (2018).
[30] L. Yang, L. Jing, J. Yu, M.K. Ng, Learning transferred weights from co-occurrence data for heterogeneous transfer learning, IEEE Trans. Neural Netw. Learn. Syst. 27 (11) (2016) 2187–2200.
[31] D.R. Hardoon, S. Szedmak, J. Shawe-Taylor, Canonical correlation analysis: An overview with application to learning methods, Neural Comput. (2004).
[32] J. Tang, X. Shu, Z. Li, G.J. Qi, J. Wang, Generalized deep transfer networks for knowledge propagation in heterogeneous domains, ACM Trans. Multimed. Comput. Commun. Appl. 12 (4) (2016) 68:1–68:22.
[33] B. Sun, K. Saenko, Deep coral: Correlation alignment for deep domain adaptation, Proceedings of the ECCV (2016).
[34] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, Domain-adversarial training of neural networks, J. Mach. Learn. Res. 17 (59) (2016) 1–35. http://jmlr.org/papers/v17/15-239.html.
[35] E. Tzeng, J. Hoffman, K. Saenko, T. Darrell, Adversarial discriminative domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7167–7176.
[36] T. Tommasi, N. Quadrianto, B. Caputo, C. Lampert, Beyond dataset bias: Multi-task unaligned shared knowledge transfer, Proceedings of the ACCV (2012).

Jianzhe Lin received the B. E. degree in Optical Engineering and the second B. A. degree in English from the Huazhong University of Science and Technology, and the master degree in the Center from Chinese Academy of Sciences, China. He is a Ph.D student in Department of Electrical and Computer Engineering at the University of British Columbia, Canada. His current research interests include computer vision and machine learning.

Liang Zhao received his M.S. and Ph.D. degrees in Software Engineering from Dalian University of Technology, China, in 2014 and 2018, respectively. He is working as a full teacher in the School of Software Technology, Dalian University of Technology. His research interests include data mining and machine learning.



Wang Qi received the B.E. degree in automation and the Ph.D. degree in pattern recognition and intelligent systems from the University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively. He is currently a Professor with the School of Computer Science and the Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xi'an, China. His research interests include computer vision and pattern recognition.



Rabab Ward is currently a Professor Emeritus with the Electrical and Computer Engineering Department,The University of British Columbia (UBC), Canada. Her research interests are mainly in the areas of signal, image,and video processing. She is a fellow of the Royal Society of Canada, the Institute of Electrical and Electronics Engineers,the Canadian Academy of Engineers,and the Engineering Institute of Canada. She is the President of the IEEE Signal Processing Society.

Z. Jane Wang received the B.Sc. degree from Tsinghua University, Beijing, China, in 1996 and the Ph.D. degree from the University of Connecticut in 2002. Since August 2004, she has been with the Department of Electrical and Computer Engineering at the University of British Columbia, Canada, where she is currently a professor.