Transfer Neural Trees for Heterogeneous Domain Adaptation

Wei-Yu Chen[1,2], Tzu-Ming Harry Hsu[2], Yao-Hung Hubert Tsai[3], Yu-Chiang Frank Wang[2], Ming-Syan Chen[1]

[1]Graduate Institute of Electrical Engineering, National Taiwan University, Taiwan
wyharveychen@gmail.com, mschen@cc.ee.ntu.edu.tw

[2]Research Center for Information Technology Innovation, Academia Sinica, Taiwan
harry19930924@gmail.com, ycwang@citi.sinica.edu.tw

[3]Department of Machine Learning, Carnegie Mellon University, USA
yaohungt@andrew.cmu.edu

Abstract. Heterogeneous domain adaptation (HDA) addresses the task of associating data not only across dissimilar domains but also described by different types of features. Inspired by the recent advances of neural networks and deep learning, we propose Transfer Neural Trees (TNT) which jointly solves cross-domain feature mapping, adaptation, and classification in a NN-based architecture. As the prediction layer in TNT, we further propose Transfer Neural Decision Forest (Transfer-NDF), which effectively adapts the neurons in TNT for adaptation by stochastic pruning. Moreover, to address semi-supervised HDA, a unique embedding loss term for preserving prediction and structural consistency between target-domain data is introduced into TNT. Experiments on classification tasks across features, datasets, and modalities successfully verify the effectiveness of our TNT.

Keywords: Transfer Learning, Domain Adaptation, Neural Decision Forest, Neural Network

1 Introduction

Domain adaptation (DA) deals with the learning tasks from data across different domains, which aims to adapt the information (e.g., labeled data) observed from different domains so that the instances in the target domain of interest can be properly described/classified [1]. A large number of computer vision and pattern recognition applications (e.g., cross-domain object recognition [2–5] and cross-language text categorization [6, 7]) can be viewed as DA problems.

Previously, approaches like [8–13] focus on associating cross-domain data described by the same type of features, which is referred to as *homogeneous* domain adaptation task. On the other hand, *heterogeneous* domain adaptation (HDA) particularly address the task of associating data not only across different domains but also in terms of distinct feature representations [14–24]. To solve the challenging HDA problems, existing approaches typically choose to determine a domain-invariant feature space, or derive a proper feature mapping for transforming cross-domain data for adaptation purposes.

Recent advances in neural networks (NN) and deep learning have shown promising results on a variety of real-world applications, including domain adaptation [10, 25–28]. However, most NN-based works for DA only consider homogeneous settings [10, 25–27]. While a recent work of [28] applied a NN architecture for associating heterogenous data across domains, cross-domain data correspondence information (i.e., co-occurrence data) is required for learning their NN. This requirement would limit its applicability for real-world HDA problems.

In this paper, we propose *Transfer Neural Trees (TNT)* as a novel NN-based architecture, which can be applied for relating and recognizing heterogeneous cross-domain data. In addition to labeled source and target-domain data, our TNT further observes unlabeled target-domain ones during adaptation, and solves semi-supervised HDA problems with improved performance. Our TNT consists of the layers of feature mapping and prediction. Without observing correspondence information across domains, the former layer is able to derive a domain-invariant intermediate representation, while a novel learner of *Transfer Neural Decision Forest (Transfer-NDF)* is obtained as the latter layer for joint adaptation and classification.

The contributions of our TNT are highlighted as follow:

- We are the first to advance neural network architectures for semi-supervised HDA, without the use of data correspondence information across domains during learning and adaptation.
- By introducing stochastic pruning, our proposed Transfer-NDF as the prediction layer in TNT is able to adapt representative neurons for relating cross-domain data.
- We uniquely advocate an embedding loss in the layer of feature mapping, which preserves the prediction and structural consistency between target-domain instances for learning TNT in a semi-supervised fashion.

2 Related work

Depending on cross-domain data described by the same/distinct types of features, homogeneous/heterogenous domain adaptation (DA) aims at associating data across domains, with the goal of bridging the knowledge between source and target domains for solving the learning tasks. When it comes to the availability of labeled instances in the target domain, one can solve homogeneous DA under *supervised* [8], *semi-supervised* [9, 13], or *unsupervised* settings [10–12].

To eliminate the domain difference, existing DA approaches either derive a domain-invariant feature subspace [8–11] for representing cross-domain data, or learn a transformation for mapping such data [13, 12]. A variety of pattern recognition tasks like object classification [11–13], text categorization [10], or speech tagging [8, 9] benefit from the recent advances of DA.

When cross-domain data are represented by different types of features, the above techniques cannot be easily extended for solving such heterogeneous domain adaptation (HDA) tasks. With the domain mismatch due to differences in terms of both data distributions and feature dimensions, it is necessary for HDA to at least obtain a limited amount of label information in the target domain, so that the adaptation across heterogeneous domains can be realized. Depending on the availability of labeled/unlabeled target-domain data, HDA approaches can be divided into supervised and semi-supervised ones.

For supervised HDA [14–19], only labeled source and (a limited amount of) target-domain data are presented during training. For example, Shi *et al.* [14] selected a fixed number of source-domain instances to perform heterogeneous spectral mapping (HeMap) for relating cross-domain data. Kulis *et al.* [15] proposed asymmetric regularized cross-domain transformation (ARC-t), which maximizes similarities on cross-domain data with identical labels. With the goal of preserving label and structure information, Wang and Mahadevan [16] solved HDA by aligning manifolds (DAMA). Inspired by [8], Duan *et al.* [17] proposed heterogeneous features augmentation (HFA) to perform common-feature-subspace learning, in which SVM classifiers were simultaneously derived. Hoffman *et al.* [18] presented a max-margin domain transformation (MMDT) to adapt SVM classifiers across domains. Similarly, Zhou *et al.* [19] considered a sparse heterogeneous feature representation (SHFR) algorithm, in which predictive structures in target domain were sparse represented by the source ones.

In contrast to supervised HDA, semi-supervised HDA [20–24] allows unlabeled target-domain data to be presented for learning and adaptation. For example, Wu *et al.* [20] proposed an algorithm of heterogeneous transfer discriminant-analysis of canonical correlations (HTDCC), which optimizes the canonical correlations of between the observed data. Li *et al.* [21] extended HFA [17] to a semi-supervised version (SHFA), with the purpose of exploiting the prediction information of unlabeled target-domain data during the learning process. By demonstrating a semi-supervised kernel matching for domain adaptation (SSKMDA), Xian and Guo [22] matched the cross-domain kernels while jointly preserving the observed data locality information. They also proposed a semi-supervised co-projection (SCP) in [23], with the objective of minimizing the divergence between cross-domain features and their prediction models. Recently, Yao *et al.* [24] presented the approach of semi-supervised domain adaptation with subspace learning (SDASL), which minimizes the prediction risk while preserving the locality structure and manifold information for HDA.

Recently, a number of researchers focus on advancing the techniques of neural networks and deep learning for solving adaptation tasks. Inspired by [10], Tzeng *et al.* [25] and Long *et al.* [29] utilized deep learning frameworks for matching distributions of cross-doman data. Both Ganin *et al.* [26] and Ajakan *et al.* [27] proposed NN-based architectures which learn classifiers for discriminating data across domains. To further handle heterogeneous cross-domain data, Shu *et al.*

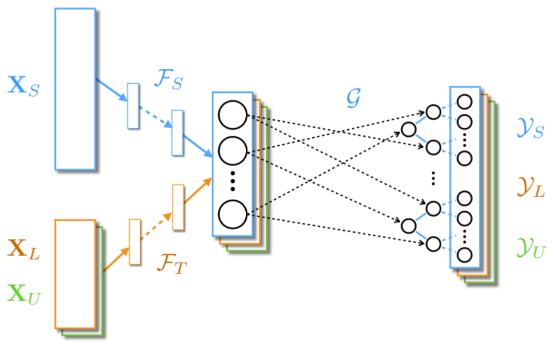

Fig. 1. The architecture of Transfer Neural Trees (TNT), which consists of the layers for feature mappings $\{\mathcal{F}_S, \mathcal{F}_T\}$ and prediction $\mathcal{G}$. To assign the labels $\mathcal{Y}_U$ for unlabeled target-domain data $\mathbf{X}_U$, TNT is learned by source-domain labeled data $\{\mathbf{X}_S, \mathcal{Y}_S\}$, together with labeled data $\{\mathbf{X}_L, \mathcal{Y}_L\}$ and the unlabeled ones $\mathbf{X}_U$ in the target domain.

[28] presented a deep neural network structure, while co-occurrence cross-domain data are required for training their networks. Based on the above observation, we propose to learn a novel NN-based framework in a semi-supervised HDA setting, without the need of co-occurrence training data pairs.

3 Proposed method

3.1 Notations and Problem Definition

For the sake of clarity, we first define the notations which will be used in the paper. We have source-domain data $\mathcal{D}_S = \{\mathbf{X}_S, \mathcal{Y}_S\} = \{\mathbf{x}_i^s, y_i^s\}_{i=1}^{n_s}$ in a d_s-dimensional space, where $\mathbf{x}_i^s \in \mathbb{R}^{d_s}$ denotes the ith source-domain instance with the corresponding label $y_i^s \in \{1, ..., C\}$. Note that n_s is the number of source-domain data, and C is the number of classes.

As for the target domain of d_t dimensions, a number n_l of labeled data $\mathcal{D}_L$ can be observed, while the remaining n_u target-domain instances $\mathcal{D}_U$ are unlabeled. Thus, we have $\mathcal{D}_L = \{\mathbf{X}_L, \mathcal{Y}_L\} = \{\mathbf{x}_i^l, y_i^l\}_{i=1}^{n_l}$, $\mathcal{D}_U = \{\mathbf{X}_U, \mathcal{Y}_U\} = \{\mathbf{x}_i^u, y_i^u\}_{i=1}^{n_u}$, and the target domain data is defined as $\mathcal{D}_T = \mathcal{D}_L \cup \mathcal{D}_U$. Note that $\mathbf{x}_i^l, \mathbf{x}_i^u \in \mathbb{R}^{d_t}$ and $y_i^l, y_i^u \in \{1, ..., C\}$. For the task of semi-supervised heterogeneous domain adaptation (HDA), the goal is to predict $\mathcal{Y}_U$ by observing $\mathcal{D}_S, \mathcal{D}_L$, and $\mathbf{X}_U$ with $n_l \ll n_u$ and $d_s \neq d_t$.

3.2 Transfer Neural Trees (TNT)

Inspired by the recent advances on neural networks and deep learning [26–28] we propose **Transfer Neural Trees (TNT)**, which can be viewed as a neural network based learning framework for solving semi-supervised HDA problems.

As illustrated in Figure 1, our TNT advocates the learning of source and target-domain mapping $\mathcal{F}_S$ and $\mathcal{F}_T$, respectively, followed by a prediction layer $\mathcal{G}$ for performing joint adaptation and classification.

It is worth noting that, our TNT enforces the mapping $\mathcal{F}_T$ to be updated in a semi-supervised fashion (i.e., $\mathcal{F}_T$ is learned by both labeled and unlabeled target-domain data). And, in order to associate and recognize cross-domain data, we propose **Transfer Neural Decision Forest (Transfer-NDF)** as the prediction layer $\mathcal{G}$ in TNT. The details of each TNT component will be discussed in the following subsections.

i) Transfer Neural Decision Forest $\mathcal{G}$ As shown in Figure 1, $\mathcal{G}$ is viewed as a prediction layer in our TNT, which is applied to adapt and recognize cross-domain data. Instead of applying existing techniques like soft-max layers and learning fine-tuned parameters for NN, we propose Transfer Neural Decision Forest (Transfer-NDF) as $\mathcal{G}$ for TNT. Benefiting from the successful developments of random forests and neural networks, our Transfer-NDF is designed to exhibit capabilities in handling and discriminating data with diverse distributions.

We now briefly review Neural Decision Forests (NDF) for the sake of completeness. Viewing decision trees as a special type of NN [30], Kontschieder *et al.* propose NDF and deep NDF (dNDF) for image classification [31, 32]. Let dNDF as a forest $\mathbf{F}$ with n_F neural decision trees, and each tree $\mathbf{T}$ in dNDF consists of $\mathcal{N}$ decision nodes and $\mathcal{L}$ leaf nodes (see Figure 2(a) for example). For an input $\mathbf{x}$ reaching a decision node $n \in \mathcal{N}$ with architecture weight/hyperplane θ_n, probabilities of d_n and $\bar{d_n} = 1 - d_n$ will be output to the subsequent nodes in the following level, i.e.,

$$d_n(\mathbf{x}; \Theta) = \sigma(f_n(\mathbf{x}; \Theta)), \tag{1}$$

with the sigmoid function $\sigma = (1 + e^{-x})^{-1}$ and $f_n(\mathbf{x}; \Theta) = \theta_n^T \mathbf{x}$ (Θ denotes the network parameter). Thus, the probability $\mu_l(\mathbf{x}|\Theta)$ at leaf node $l \in \mathcal{L}$ is:

$$\mu_l(\mathbf{x}|\Theta) = \prod_{n \in \mathcal{N}} d_n(\mathbf{x}; \Theta)^{\mathbf{1}_{l \swarrow n}} \bar{d_n}(\mathbf{x}; \Theta)^{\mathbf{1}_{l \searrow n}}. \tag{2}$$

Note that $\mathbf{1}_{l \swarrow n}$ and $\mathbf{1}_{l \searrow n}$ indicate the decision at node n when traversing a path along $\mathbf{T}$ to reach the leaf node l.

In dNDF, each leaf node observes class-label distribution $\boldsymbol{\pi}_l = \{\pi_{l_1}, ..., \pi_{l_C}\}$ with each entry denoting the probability of taking the corresponding class. Thus, the prediction from all leaf nodes in $\mathbf{T}$ is computed as:

$$\mathbb{P}_{\mathbf{T}}[y|\mathbf{x}, \Theta, \boldsymbol{\pi}] = \sum_{l \in \mathcal{L}} \pi_{l_y} \mu_l(\mathbf{x}|\Theta). \tag{3}$$

Finally, the overall prediction from the entire forest $\mathbf{F}$ is determined by:

$$\mathbb{P}_{\mathbf{F}}[y|\mathbf{x}, \Theta, \boldsymbol{\pi}] = \frac{1}{n_F} \sum_{\mathbf{T} \in \mathbf{F}} \mathbb{P}_{\mathbf{T}}[y|\mathbf{x}, \Theta, \boldsymbol{\pi}]. \tag{4}$$

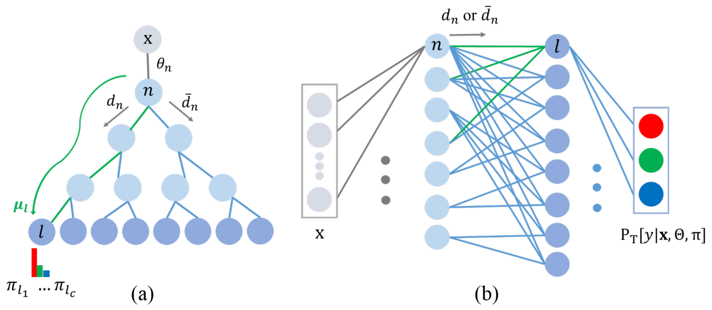

Fig. 2. (a) Illustration of each tree structure in dNDF and (b) visualization of dNDF in terms of a neural network architecture.

As noted in [32], the overall objective function of dNDF is determined by the log-loss term for the training/labeled data, i.e.,

$$L_p(\Theta, \boldsymbol{\pi}; \mathbf{x}, y) = \sum_{\mathbf{T} \in \mathbf{F}} -\log(\mathbb{P}_{\mathbf{T}}[y|\mathbf{x}, \Theta, \boldsymbol{\pi}]). \tag{5}$$

With stochastic routing and differentiable properties, each decision and leaf node in dNDF can be learned via back propagation, which makes dNDF as an effective alternative to (deep) NN when learning the network parameters.

Despite the above promising properties, dNDF *cannot* be easily extended for domain adaptation tasks. This is because that, not all the leaf nodes learned by source-domain data can generalize to describing target-domain data. More precisely, if a leaf node is seldom updated by source-domain data, the corresponding class-label distribution might not be sufficiently representative for adaptation.

Instead of selecting a threshold to prune such leaf nodes and resulting in a complex neural network in Figure 2(b), we introduce a novel *stochastic pruning* approach for learning our Transfer-NDF. Given an input $\mathbf{x}$ and network Θ, we choose to learn and update the class-label distribution at a leaf node l by:

$$\pi_{l_y}^{(t+1)} = \frac{1}{Z_l^{(t)}}(p_d + \sum_{(\mathbf{x},y') \in \mathcal{D}_S} \frac{\mathbf{1}_{y=y'} \pi_{l_y}^{(t)} \mu_l(\mathbf{x}|\Theta)}{\mathbb{P}_{\mathbf{T}}[y|\mathbf{x}, \Theta, \boldsymbol{\pi}^{(t)}]}). \tag{6}$$

In (6), μ_l is the probability of the input $\mathbf{x}$ reaching leaf node l, $\boldsymbol{\pi}_l^{(t)}$ is the derived class-label distribution at iteration t, and $\mathbb{P}_{\mathbf{T}}$ denotes the probability of taking class y. Note that $\mathbf{1}$ is the indicator function, and Z_l is a normalizing factor ensuring that $\sum_y \pi_{l_y}^{(t+1)} = 1$. Different from dNDF, we choose to add a small positive p_d in (6) for updating the class-label distributions of *each* leaf node.

We now explain why the use of (6) in our Transfer-NDF can be viewed as a stochastic pruning technique and thus is preferable for domain adaptation. We note that, the second term in (6) (also presented in the original dNDF definition) counts the number of $\mathbf{X}_S$ reaching leaf node l with decision y. If a leaf node is

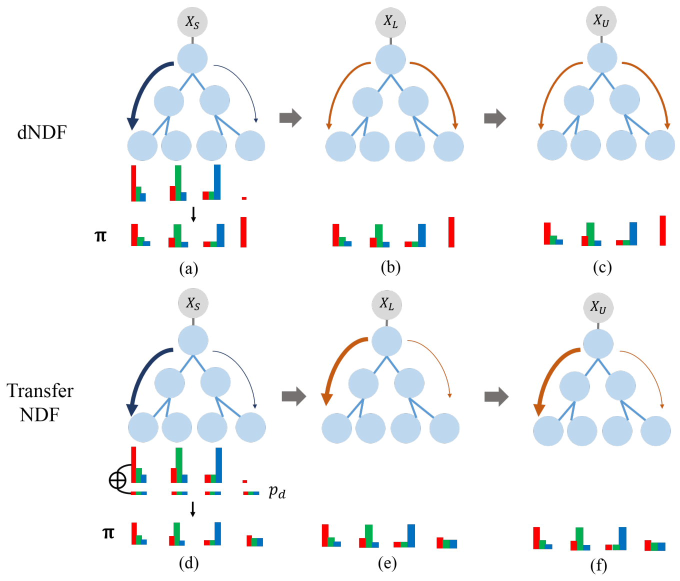

Fig. 3. Comparisons of dNDF [32] and Transfer-NDF as the decision layer $\mathcal{G}$ in TNT. Learning class-label distributions $\boldsymbol{\pi}$ using $\mathbf{X}_S$, adaptation of $\mathbf{X}_L$, and prediction of $\mathbf{X}_U$ using dNDF are shown in (a), (b), and (c), respectively. The associated processes with Transfer-NDF are shown in (d), (e), and (f).

only updated by few source-domain instances, the resulting distribution would be sparse and with small π_{l_i} values. However, the normalization process in (6) would amplify their distribution values, as illustrated in the resulting $\boldsymbol{\pi}$ of the rightmost leaf node in Figure 3(a). When $\mathbf{X}_L$ are taken as inputs, the prediction loss observed from all leaf nodes would be considered as equally important when updating TNT via back propagation. As a result, prediction of the unlabeled ones $\mathbf{X}_U$ would overfit the outputs from such leaf nodes (see Figure 3(c)).

With the introduction of p_d in (6), we are able to suppress the above extreme class-label distributions at the leaf nodes with seldom updates. As illustrated in Figures 3(d), adding p_d in (6) would turn the class-label distributions of such nodes close to uniform distribution after normalization. Thus, as depicted in Figures 3(e), no strong prediction results can be inferred by $\mathbf{X}_L$ reaching such leaf nodes. This allows the prediction of $\mathbf{X}_U$ to be dominated by the leaf nodes with sufficient representation ability only (see Figure 3(f)). It is worth repeating that, with this stochastic pruning process for constructing TNT, we do not need to carefully select and disregard particular nodes for adaptation purposes.

ii) Feature Mapping $\mathcal{F}_S$ and $\mathcal{F}_T$ As illustrated in Figure 1, $\mathcal{F}_S$ and $\mathcal{F}_T$ in our TNT are neural network based structures which map source and target-domain data for representation learning. Once the domain-invariant feature representation is derived, the prediction layer $\mathcal{G}$ (i.e., Transfer-NDF) will be applied for joint adaptation and classification. While $\mathcal{F}_S$ and $\mathcal{F}_T$ share the same goal of learning cross-domain feature representation, we need to derive these two mappings separately using the observed data in the associated domain. Moreover, when learning $\mathcal{F}_T$, we will utilize both labeled and unlabeled target-domain data for learning our TNT in a semi-supervised setting.

In our work, we apply hyperbolic tangent as the activation function for $\mathcal{F}_S$. When observing the source-domain data as inputs, we have (5) as the objective function with back propagation to update $\mathcal{F}_S$ (and $\mathcal{G}$), with the goal of minimizing the prediction error observed from all the leaf nodes. Similar remarks can be applied to the update of $\mathcal{F}_T$ using labeled target-domain data $\mathbf{X}_L$.

However, due to the lack of label information, we cannot apply (5) for learning $\mathcal{F}_T$ with unlabeled target-domain data $\mathbf{X}_U$. In our work, we advocate to preserve the prediction and structural consistency between $\mathbf{X}_L$ and $\mathbf{X}_U$, and propose to enforce an *embedding loss term* L_e defined as follows:

$$\begin{aligned} L_e(\Theta, \boldsymbol{\pi}; \mathbf{x}, \tilde{y}) &= \sum_{\mathbf{x}\in\{\mathbf{X}_L, \mathbf{X}_U\}, \mathbf{T}\in\mathbf{F}} -\mathbb{P}_{\mathbf{T}}[\tilde{y}|\mathbf{x}, \Theta, \boldsymbol{\pi}] \frac{\mathbb{P}_{\mathbf{F}}[\tilde{y}|\mathbf{x}, \Theta, \boldsymbol{\pi}]}{\mathbb{P}_{\mathbf{F}}[\tilde{y}|\Theta, \boldsymbol{\pi}]}, \\ \mathbb{P}_{\mathbf{F}}[\tilde{y}|\Theta, \boldsymbol{\pi}] &= \frac{1}{n_l + n_u} \sum_{\mathbf{x}\in\{\mathbf{X}_L, \mathbf{X}_U\}} \mathbb{P}_{\mathbf{F}}[\tilde{y}|\mathbf{x}, \Theta, \boldsymbol{\pi}]. \end{aligned} \tag{7}$$

In (7), $\tilde{y}$ denotes the output labels of inputs $\mathbf{x} \in \{\mathbf{X}_L, \mathbf{X}_U\}$, and $\mathbb{P}_{\mathbf{F}}[\tilde{y}|\Theta, \boldsymbol{\pi}]$ is viewed as a normalization term. We can see that, minimizing this embedding loss term L_e is equivalent to maximizing the prediction consistency between the output of each tree $\mathbb{P}_{\mathbf{T}}[\tilde{y}|\mathbf{x}, \Theta, \boldsymbol{\pi}]$ and that of the decision forests $\mathbb{P}_{\mathbf{F}}[\tilde{y}|\mathbf{x}, \Theta, \boldsymbol{\pi}]$. By performing adaptation and classification using the prediction layer $\mathcal{G}$ in our TNT, the above process further implies that the structural consistency between $\mathbf{X}_L$ and $\mathbf{X}_U$ can be preserved.

It is worth noting that, when updating $\mathcal{F}_T$, the decision layer $\mathcal{G}$ (i.e., Transfer-NDF) learned from source-domain data is remained fixed. This is to enforce and to adapt $\mathcal{G}$ for recognizing target-domain data. The details of the above learning process will be discussed in the next subsection.

3.3 Learning of TNT

i) Learning from Source-Domain Data When observing source-domain data $\mathcal{D}_S = \{\mathbf{X}_S, \mathcal{Y}_S\}$ for learning our TNT, both mapping $\mathcal{F}_S$ and the decision layer $\mathcal{G}$ (i.e., Transfer-NDF) will be updated by:

$$\min_{\mathcal{F}_S, \mathcal{G}} \sum_{(\mathbf{x}, y)\in\mathcal{D}_S} L_p(\Theta, \boldsymbol{\pi}; \mathbf{x}, y). \tag{8}$$

As determined in (5), L_p returns the prediction loss of the input labeled data. We take the derivative of L_p with respect to Θ, and apply back propagation to update the architectures of $\mathcal{F}_S$ and $\mathcal{G}$:

$$\frac{\partial L_p}{\partial \Theta}(\Theta, \boldsymbol{\pi}; \mathbf{x}, y) = \sum_{n\in\mathcal{N}} \frac{\partial L_p(\Theta, \boldsymbol{\pi}; \mathbf{x}, y)}{\partial f_n(\mathcal{F}(\mathbf{x}), \Theta)} \frac{\partial f_n(\mathcal{F}(\mathbf{x}), \Theta)}{\partial \Theta}, \tag{9}$$

where $\mathcal{F}$ indicates $\mathcal{F}_S$ or $\mathcal{F}_T$, and we have

$$\frac{\partial f_n(\mathcal{F}(\mathbf{x}),\Theta)}{\partial \Theta} = \frac{\partial \theta_n^T \mathcal{F}(\mathbf{x})}{\partial \Theta} = \mathcal{F}(\mathbf{x}) \text{ and}$$
$$\frac{\partial L_p(\Theta,\boldsymbol{\pi};\mathbf{x},y)}{\partial f_n(\mathcal{F}(\mathbf{x}),\Theta)} = \frac{\sum_{l\in\mathcal{L}} \mathbf{1}_{l\searrow n} d_n(\mathbf{x};\Theta)\boldsymbol{\pi}_{l_y}\mu_l(\mathbf{x}|\Theta) - \mathbf{1}_{l\swarrow n}\bar{d}_n(\mathbf{x};\Theta)\boldsymbol{\pi}_{l_y}\mu_l(\mathbf{x}|\Theta)}{\mathbb{P}_{\mathbf{T}}[y|\mathbf{x},\Theta,\boldsymbol{\pi}]}. \tag{10}$$

The details of the above derivations can be found in [32].

ii) Learning from Target-Domain Data Once $\mathcal{F}_S$ and $\mathcal{G}$ are obtained, we have $\mathbf{X}_L$ and $\mathbf{X}_U$ as the target-domain inputs for deriving $\mathcal{F}_T$. Recall that $\mathcal{G}$ is only updated by source-domain data. Since it is utilized to adapt target-domain data, it would remain fixed when we learn the mapping $\mathcal{F}_T$.

For the mapping $\mathcal{F}_T$, we choose to solve the following optimization task:

$$\min_{\mathcal{F}_T} \sum_{(\mathbf{x},y)\in\mathcal{D}_L} L_p(\Theta,\boldsymbol{\pi};\mathbf{x},y) + \lambda \sum_{\tilde{y}} \sum_{\mathbf{x}\in\{\mathbf{X}_L,\mathbf{X}_U\}} L_e(\Theta,\boldsymbol{\pi};\mathbf{x},\tilde{y}). \tag{11}$$

Note that the first term in (11) is in the same formulation as (8), which only calculates the loss observed from the labeled target-domain data $\mathbf{X}_L$. As introduced in Section 3.2, the second term L_e is the *embedding loss term* determined in (7) (with $\tilde{y}$ denoting the predicted labels). As discussed earlier, this L_e term enforces prediction/structural consistency between target-domain data, so that semi-supervised HDA can be addressed accordingly.

In (11), we have parameter λ regularizing the embedding loss term. As suggested by [26], we gradually increase the value of λ during iterative updates. This is because that the loss produced by $\mathbf{X}_L$ in early iterations are not sufficiently reliable, and thus we do not emphasize the feedback of L_e in the early stage of our TNT derivation.

To update $\mathcal{F}_T$ via back propagation, we also take the derivatives of L_p and L_e with respect to Θ. While the derivative of L_p is in the same form as (9), that of L_e needs additional efforts to calculate its partial derivative with respect to f_n (see Supplementary for detailed derivations), i.e.,

$$\frac{\partial L_e(\Theta,\boldsymbol{\pi};\mathbf{x},\tilde{y})}{\partial f_n(\mathcal{F}(\mathbf{x});\Theta)} = \frac{\partial L_e(\Theta,\boldsymbol{\pi};\mathbf{x},\tilde{y})}{\partial \mathbb{P}_{\mathbf{T}}[\tilde{y}|\mathbf{x},\Theta,\boldsymbol{\pi}]} \frac{\partial \mathbb{P}_{\mathbf{T}}[\tilde{y}|\mathbf{x},\Theta,\boldsymbol{\pi}]}{\partial f_n(\mathcal{F}(\mathbf{x});\Theta)}$$
$$\approx \frac{\mathbb{P}_{\mathbf{F}}[\tilde{y}|\mathbf{x},\Theta,\boldsymbol{\pi}]}{\mathbb{P}_{\mathbf{F}}[\tilde{y}|\Theta,\boldsymbol{\pi}]} \sum_{l\in\mathcal{L}} \mathbf{1}_{l\searrow n} d_n(\mathbf{x};\Theta)\boldsymbol{\pi}_{l_y}\mu_l(\mathbf{x}|\Theta) - \mathbf{1}_{l\swarrow n}\bar{d}_n(\mathbf{x};\Theta)\boldsymbol{\pi}_{l_y}\mu_l(\mathbf{x}|\Theta). \tag{12}$$

It is worth noting that, despite the learning of $\mathcal{F}_T$ enforces the prediction consistency between target-domain data, the diversity of all trees in Transfer-NDF $\mathcal{G}$ can be preserved. This is because that the prediction layer $\mathcal{G}$ in TNT is fixed when learning $\mathcal{F}_T$. Thus, the resulting adaptation/recogntion performance would not be affected.

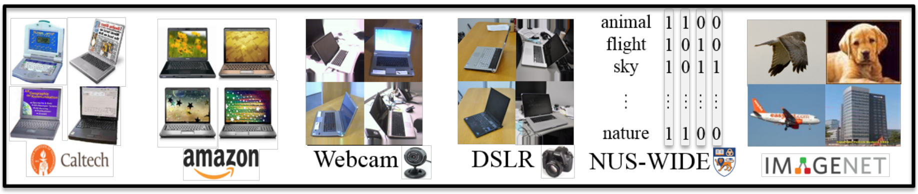

Fig. 4. Cross-domain datasets: **Caltech-256**, **Office**, **NUS-WIDE** and **ImageNet**. Note that we apply **Caltech-256** and **Office** for cross-domain object recognition, and we perform translated learning using **NUS-WIDE** and **ImageNet**.

4 Experiment

4.1 Datasets and Settings

We consider cross-domain classification tasks in our experiments. We first address cross-domain object recognition problems using the datasets of **Office** + **Caltech-256** datasets [33, 2]. The former is composed of object images of 31 categories, collected from three sources: Amazon (A), Webcam (W) and DSLR (D). On the other hand, **Caltech-256** contains 256 object categories also collected from the Internet. Among these objects, 10 overlapping categories are considered for experiments. For HDA tasks, we consider two type of features: DeCAF$_6$ [34] and SURF [35]. The former is of 4096 dimensions, while the latter is a 100-dimensional bag-of-word feature representation.

In additional to cross-domain object recognition, we further consider the task of associating and recognizing text and image data, also referred to as *translated learning* [6]. While most existing works on this task requires co-occurrence text and image data for learning purposes, we will show that our TNT is able to solve this cross-domain classification problem in a semi-supervised setting without the need of any cross-domain co-occurrence data. Following the setting of [28], we apply NUS-WIDE [36] and ImageNet [37] as the datasets for text and images, respectively. NUS-WIDE contains of tag information of 269,648 Flickr images, while ImageNet is with 5247 synsets and 3.2 million images in total. The selected datasets and their examples images are shown in Figure 4.

To preprocess the NUS-WIDE tag data, we pre-trained a 5-layer NN with a soft-max layer, and take the 4th hidden layer as the 64-dimensional feature representation. As for the ImageNet data, we follow [38] and extract their DeCAF$_6$ features as representation. For simplicity, we choose 8 overlapping categories of these two datasets: *airplane, birds, buildings, cars, dog, fish, flowers, horses.*

For fair comparisons, we fix our TNT settings for all experiments. Both $\mathcal{F}_S$ and $\mathcal{F}_T$ are single-layer neural networks, which apply hyperbolic tangent as the activation function, with the dimension of the mapping output as 100. Transfer-NDF $\mathcal{G}$ is composed of 20 trees with depth of 7, and each tree randomly samples 20 dimensions from the mapping output (out of 100). And, we have p_d fixed as 0.001, and λ is gradually increased from 0 to 0.1 as noted in Section 3.3.

Table 1. Performance comparisons for cross-feature object recognition. Note that $\mathcal{D}_S$ and $\mathcal{D}_T$ denote source (DeCAF$_6$) and target (SURF) domains, respectively.

$\mathcal{D}_S \rightarrow \mathcal{D}_T$	SVM$_t$	NN$_t$	MMDT [18]	HFA [21]	SHFR [19]	SCP [23]	TNT
A → A	45.37	45.80	45.47	46.66	44.50	45.47	**50.41**
C → C	37.15	35.02	35.50	36.32	33.39	34.67	**39.03**
W → W	61.51	61.06	61.13	61.89	54.34	60.00	**62.34**

Table 2. Performance comparisons using NN, dNDF, and Transfer-NDF as $\mathcal{G}$ in TNT. Note that Transfer-NDF* denotes dNDF with our stochastic pruning, while our Tranfer-NDF observes both labeled and unlabeled target-domain data during adaptation by introducing L_e in (11).

$\mathcal{D}_S \rightarrow \mathcal{D}_T$	NN	dNDF	Transfer-NDF*	Transfer-NDF
C → A	42.84	46.57	46.72	**49.50**
C → C	32.31	33.94	34.62	**39.03**
C → W	56.75	60.30	61.36	**62.42**

4.2 Evaluation

When comparing the performance of different HDA approaches, we first consider SVM$_t$ and NN$_t$ as baseline methods, which simply take labeled target-domain data for training SVM and (two-layer) NN, respectively. As for the state-of-the-art HDA ones, we include the results produced by MMDT [18], HFA [21], SCP [23] and SHFR [19] for comparisons.

Object recognition across feature spaces For the task of object recognition, we first address the problem of cross-feature image recognition using Office + Caltech-256. DSLR (D) is excluded in the experiments, since only a limited amount of data is available for this subset. For source-domain data, we take images in CNN features (i.e., DeCAF$_6$); as for the target-domain images, we have SURF as the features. In the semi-supervised settings of HDA, we randomly choose 3 images in the target domain as labeled data, while the remaining one in that domain as the images to be recognized.

Table 1 lists the average classication results with 5 random trials. From this table, we see that all the HDA approaches were able to produce improved performance when comparing to SVM$_t$ and NN$_t$. And, our TNT consistently achieves the best results among all cross-feature pairs in this experiment.

Moreover, we further verify the use of our Transfer-NDF in TNT for associating and recognizing cross-domain data. In Table 2, we compare the results of using NN, dNDF and our Transfer-NDF as $\mathcal{G}$ in TNT (with DeCAF$_6$ and SURF describing source and target-domain data, respectively). We can see that, while the use of dNDF exhibited improved adaptation ability than NN, introducing stochastic pruning to dNDF (i.e., Transfer-NDF* in Table 2) further increased the recognition performance. Nevertheless, the full version of Transfer-NDF was able to achieve the best performance.

Table 3. Performance comparisons of object recognition across features and datasets. Note that DeCAF$_6$ and SURF describe source and target-domain data, respectively.

$\mathcal{D}_S \rightarrow \mathcal{D}_T$	SVM$_t$	NN$_t$	MMDT [18]	HFA [21]	SHFR [19]	SCP [23]	TNT
C → A	45.37	45.80	45.69	46.44	44.61	41.59	**49.50**
W → A			46.23	46.98	43.86	44.50	**48.45**
A → C	37.15	35.02	35.77	36.32	33.39	35.04	**38.37**
W → C			36.05	36.41	33.21	35.96	**37.75**
A → W	61.51	61.06	61.13	**61.89**	54.34	58.87	60.08
C → W			60.76	62.26	54.34	51.32	**62.42**

Table 4. Comparisons of classification results by adapting text (NUS-WIDE) to image data (ImageNet).

$\mathcal{D}_S \rightarrow \mathcal{D}_T$	SVM$_t$	NN$_t$	MMDT [18]	HFA [21]	SHFR [19]	SCP [23]	TNT
tag → image	63.75	65.65	49.38	62.88	60.38	65.00	**72.22**

Object recognition across features and datasets We now consider a more challenging object recognition task, in which source and target domains observe distinct features (i.e, DeCAF$_6$ vs. SURF) and from different datasets (i.e., A, C, or W). The average recognition results of different methods are listed in Table 3. From this table, we see that our TNT produced comparable or improved results than baseline and recent HDA approaches did, which confirms that the use of our TNT for cross-domain object recognition is preferable.

Text-to-image classification As noted earlier, we further consider the adaptation and classification of text and image data. With tag information observed in the source domain dataset of NUS-WIDE, our goal is to improve image classification using the ImageNet data. For the semi-supervised setting in the target domain, we randomly select 3 images per category as labeled data, and 100 images for prediction.

Table 4 lists and compares the performances of different approaches with 5 random trials. It can be seen that, without utilizing co-occurrence data across domains for learning, it is not an easy task for solving such cross-domain classification problems. Our TNT was able to achieve a significantly improved result under this semi-supervised setting, and this shows the effectiveness and robustness of our TNT for handling heterogeneous cross-domain data.

4.3 Analysis and Visualization

Adaptation ability of Transfer-NDF Recall that, in Table 2, we compare the performance of the uses of different classifiers as the prediction layer $\mathcal{G}$ in TNT. We now visualize the observed cross-data distributions at the mapping layer output (i.e., the input layer of $\mathcal{G}$), and show the results in Figure 5.

Comparing Figures 5(a) and (b), we see the use of NN in TNT over-fitted the target-domain labeled data, and such problems were only slightly alleviated when dNDF was applied as $\mathcal{G}$. While the supervised version of TNT (i.e., Transfer-NDF*) further improved the adaptation ability as depicted in Figure 5(c), the

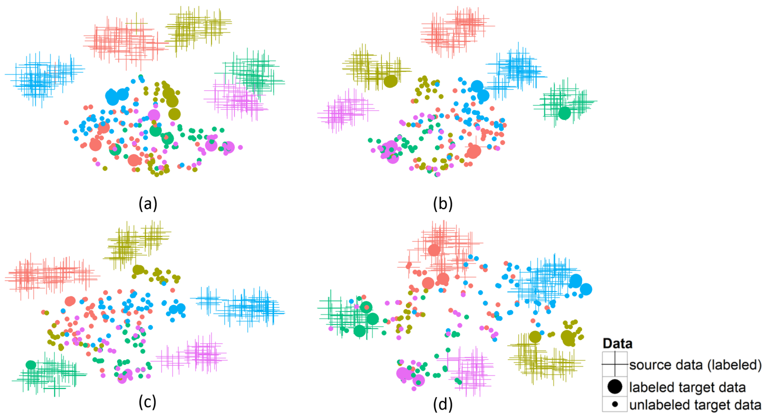

Fig. 5. t-SNE visualization of cross-domain data distributions using NN, dNDF, and Transfer-NDF as $\mathcal{G}$ in TNT: (a) NN, (b) dNDF, (c) Transfer-NDF*, and (d) Transfer-NDF. As noted in Table 2, Transfer-NDF* denotes dNDF with stochastic pruning.

full version of our Transfer-NDF successfully related cross-domain heterogenous data in this layer and achieved the best recognition performance.

Stochastic pruning for Transfer-NDF To confirm that our introduced stochastic pruning for Transfer-NDF would disregard the leaf nodes with insufficient adaptation ability, We plot the observed class-label distributions $\boldsymbol{\pi}$ without and with such pruning (i.e., dNFD and Transfer-NDF) in Figures 6(a) and (b), respectively. Note that each row in Figure 6(a) or (b) denotes a tree, in which each entry indicates a leaf node. The color (out of 10 categories) of each entry denotes the dominant class label observed from the associated distribution.

In Figure 6(a), we see that dNDF (i.e., no stochastic pruning) produced the leaf nodes with different colors. This means that all the leaf nodes were considered equally important for adaptation/prediction, as discussed in Section 3.2. With stochastic pruning (i.e., adding a small constant $p_d = 0.001$ for all class labels in (6)), a portion of the leaf nodes were not able to observe dominant distribution values, and thus were shown in black in Figure 6(b). Note that a cutoff threshold is set as 0.3 in Figure 6(b) for visualization purposes. Thus, in addition the quantitative evaluation, the above observation further supports our Transfer-NDF for adapting cross-domain data.

Enforcing embedding loss L_e Finally, we verify the effectiveness of introducing L_e in TNT for learning $\mathcal{F}_T$ in a semi-supervised setting. Using Transfer-NDF without and with L_e (i.e., Transfer-NDF* and Transfer-NDF, respectively), we observe the associated variances of the predicted probability outputs of all the target-domain instances, and we plot the results in Figure 7.

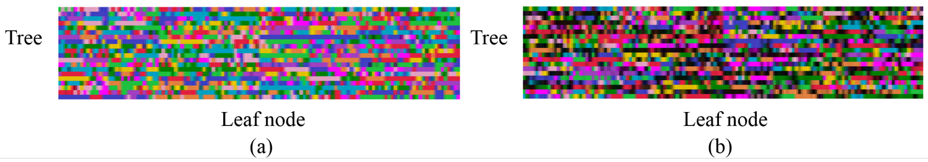

Fig. 6. Class-label distribution $\boldsymbol{\pi}$ of (a) dNDF and (b) Transfer-NDF on C (DeCAF$_6$) $\rightarrow$ C (SURF). Note that each entry indicates the distribution value of the dominant class (out of 10 categories/colors).

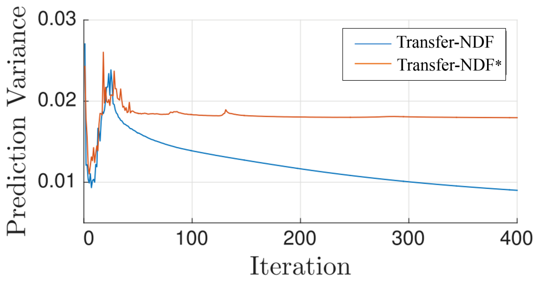

Fig. 7. Prediction variance vs. iteration number on C (DeCAF$_6$) $\rightarrow$ C (SURF). Note that Transfer-NDF* (i.e., dNDF with stochastic pruning) is learned from labeled cross-domain data only.

Comparing the two curves in Figure 7, we see that the use of L_e (i.e., Transfer-NDF) was able to reduce the variance of the prediction probability outputs during the adaptation process. This supports our argument in Section 3.3 that, the enforcement of prediction consistency between labeled and unlabeled target-domain data would preserve their structural consistency during adaptation. This is the reason why our TNT is able to handle HDA in a semi-supervised setting.

5 Conclusion

We presented Transfer Neural Trees (TNT) for adapting and recognizing cross-domain heterogeneous data in a semi-supervised setting. With the embedding loss for enforcing prediction and structural consistency between target-domain data, plus the use of our Transfer-NDF with stochastic pruning for adapting representative neurons, our TNT is able to solve feature mapping, adaptation, and classification in a unified NN-based framework. Our experiments confirmed that the proposed TNT performed favorably against state-of-the-art HDA approaches on a variety of classification tasks using data across different feature spaces, collected by different datasets, or in terms of distinct modalities. Among the future research directions, one can extend the feature mapping layers in our TNT architecture. This would further allow the learning and adaptation of cross-domain data with increasing sizes or modalities with more complexities.

References

1. Pan, S.J., Yang, Q.: A survey on transfer learning. In: IEEE Transactions on Knowledge and Data Engineering. (2010)
2. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: ECCV. (2010)
3. Zhu, Y., Chen, Y., Lu, Z., Pan, S.J., Xue, G.R., Yu, Y., Yang, Q.: Heterogeneous transfer learning for image classification. In: AAAI. (2011)
4. Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Simultaneous deep transfer across domains and tasks. In: IEEE ICCV. (2015)
5. Chidlovskii, B., Csurka, G., Gangwar, S.: Assembling heterogeneous domain adaptation methods for image classification. In: CLEF (Working Notes). (2014)
6. Dai, W., Chen, Y., Xue, G.R., Yang, Q., Yu, Y.: Translated learning: Transfer learning across different feature spaces. In: NIPS. (2008)
7. Prettenhofer, P., Stein, B.: Cross-language text classification using structural correspondence learning. In: ACL. (2010)
8. Daumé III, H.: Frustratingly easy domain adaptation. In: ACL. (2007)
9. Daumé III, H., Kumar, A., Saha, A.: Frustratingly easy semi-supervised domain adaptation. In: Natural Language Processing Workshop. (2010)
10. Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q.: Domain adaptation via transfer component analysis. In: IEEE Transactions on Neural Networks. (2011)
11. Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised domain adaptation. In: IEEE CVPR. (2012)
12. Fernando, B., Habrard, A., Sebban, M., Tuytelaars, T.: Unsupervised visual domain adaptation using subspace alignment. In: IEEE ICCV. (2013)
13. Donahue, J., Hoffman, J., Rodner, E., Saenko, K., Darrell, T.: Semi-supervised domain adaptation with instance constraints. In: IEEE CVPR. (2013)
14. Shi, X., Liu, Q., Fan, W., Yu, P.S., Zhu, R.: Transfer learning on heterogenous feature spaces via spectral transformation. In: IEEE ICDM. (2010)
15. Kulis, B., Saenko, K., Darrell, T.: What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In: IEEE CVPR. (2011)
16. Wang, C., Mahadevan, S.: Heterogeneous domain adaptation using manifold alignment. In: IJCAI. (2011)
17. Duan, L., Xu, D., Tsang, I.: Learning with augmented features for heterogeneous domain adaptation. In: ICML. (2012)
18. Hoffman, J., Rodner, E., Donahue, J., Darrell, T., Saenko, K.: Efficient learning of domain-invariant image representations. In: ICLR. (2013)
19. Zhou, J.T., Tsang, I.W., Pan, S.J., Tan, M.: Heterogeneous domain adaptation for multiple classes. In: AISTATS. (2014)
20. Wu, X., Wang, H., Liu, C., Jia, Y.: Cross-view action recognition over heterogeneous feature spaces. In: IEEE ICCV. (2013)
21. Li, W., Duan, L., Xu, D., Tsang, I.W.: Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation. In: IEEE T-PAMI. (2014)
22. Xiao, M., Guo, Y.: Feature space independent semi-supervised domain adaptation via kernel matching. In: IEEE T-PAMI. (2015)
23. Xiao, M., Guo, Y.: Semi-supervised subspace co-projection for multi-class heterogeneous domain adaptation. In: ECML. (2015)
24. Yao, T., Pan, Y., Ngo, C.W., Li, H., Mei, T.: Semi-supervised domain adaptation with subspace learning for visual recognition. In: IEEE CVPR. (2015)

25. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., Darrell, T.: Deep domain confusion: Maximizing for domain invariance. In: CoRR, abs/1412.3474. (2014)
26. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: ICML. (2015)
27. Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M.: Domain-adversarial neural networks. In: JMLR. (2014)
28. Shu, X., Qi, G.J., Tang, J., Wang, J.: Weakly-shared deep transfer networks for heterogeneous-domain knowledge propagation. In: ACM Conference on Multimedia Conference. (2015)
29. Long, M., Wang, J.: Learning transferable features with deep adaptation networks. In: ICML. (2015)
30. Sethi, I.K.: Entropy nets: from decision trees to neural networks. In: Proceedings of the IEEE (Special Issue on Neural Networks). (1990)
31. Rota Bulo, S., Kontschieder, P.: Neural decision forests for semantic image labelling. In: IEEE CVPR. (2014)
32. Kontschieder, P., Fiterau, M., Criminisi, A., Rota Bulo, S.: Deep neural decision forests. In: IEEE ICCV. (2015)
33. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. (2007)
34. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: ICML. (2014)
35. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: ECCV. (2006)
36. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: Nus-wide: a real-world web image database from national university of singapore. In: ACM international conference on image and video retrieval. (2009)
37. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: IEEE CVPR. (2009)
38. Tommasi, T., Tuytelaars, T.: A testbed for cross-dataset analysis. In: ECCV Workshops. (2014)