PRACTICALLY EASY TRANSFER LEARNING

Jindong Wang[1], Yiqiang Chen[1,⋆], Han Yu[2], Meiyu Huang[3], Qiang Yang[4]

[1]Beijing Key Lab. of Mobile Computing and Pervasive Device, Inst. of Computing Tech., CAS
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[3]Qian Xuesen Lab. of Space Tech., China Academy of Space Tech., Beijing, China
[4]Hong Kong University of Science and Technology, Hong Kong, China
{wangjindong, yqchen}@ict.ac.cn, han.yu@ntu.edu.sg, huangmeiyu@qxslab.cn, qyang@cse.ust.hk

ABSTRACT

Transfer learning aims at transferring knowledge from a well-labeled domain to a similar but different domain with limited or no labels. Unfortunately, existing learning-based methods often involve intensive model selection and hyperparameter tuning to obtain good results. Moreover, cross-validation is not possible for tuning hyperparameters since there are often no labels in the target domain. This would restrict wide applicability of transfer learning especially in computationally-constraint devices such as wearables. In this paper, we propose a practically *Easy Transfer Learning (EasyTL)* approach which requires no model selection and hyperparameter tuning, while achieving competitive performance. By exploiting intra-domain structures, EasyTL is able to learn both non-parametric transfer features and classifiers. Extensive experiments demonstrate that, compared to state-of-the-art traditional and deep methods, EasyTL satisfies the *Occam's Razor* principle: it is extremely easy to implement and use while achieving comparable or better performance in classification accuracy and much better computational efficiency. Additionally, it is shown that EasyTL can increase the performance of existing transfer feature learning methods.

Index Terms— Transfer Learning, Domain Adaptation, Cross-domain Learning, Non-parametric Learning

1. INTRODUCTION

The success of multimedia applications depends on the availability of sufficient labeled data to train machine learning models. However, it is often expensive and time-consuming to acquire massive amounts of labeled data. Transfer learning (TL), or domain adaptation [1] is a promising strategy to enhance the learning performance on a target domain with few or none labels by leveraging knowledge from a well-labeled source domain. Since the source and target domains have different distributions, numerous methods have been proposed to reduce the distribution divergence [2, 3, 4, 5, 6].

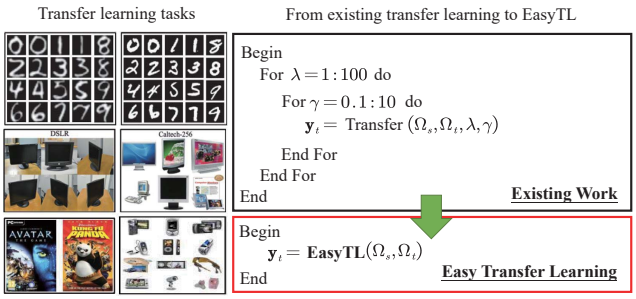


Fig. 1. A brief illustration of EasyTL

Table 1. Comparison between EasyTL and recent TL methods on Office-Home [7] dataset

Method	MEDA [3]	DANN [8]	CDAN [9]	EasyTL
Accuracy	60.3	57.6	62.8	**63.3**
Hyperparameter	λ, η, p	λ	λ	**None**
Network	ResNet50 [10]			

Unfortunately, despite the great performance achieved by existing TL methods, it is notoriously challenging to apply them to a real situation since we cannot determine the best TL model and their optimal hyperparameters. The reasons are three-fold. Firstly, most of the traditional and deep TL methods are *parametric* methods [3, 2, 5, 8] that have to go through an intensively expensive and time-consuming process to tune a lot of hyperparameters (e.g. Fig. 1, MEDA, DANN, and CDAN in Table 1). Secondly, *cross-validation*, which is the most common strategy to select models and tune hyperparameters, is *not available* in TL since there are often no labeled data in the target domain [1]. Thirdly, although the recent AutoML algorithms can automatically tune the hyperparameters via tree pruning, boosting, or neural architecture search [11], they are incapable of handling the different distributions between domains in TL and typically take a long time to converge.

These challenges seriously restrict the real application of TL, especially on small devices that require instant local computing with limited computational resources such as wear-

Corresponding author: Y. Chen. J. Wang and Y. Chen are also affiliated with University of Chinese Academy of Sciences, Beijing 100190, China.
978-1-5386-1737-3/18/$31.00 2019 IEEE.

ables. Is it possible to develop an easy but powerful TL algorithm to circumvent model selection and hyperparameter tuning, but have competitive performance, i.e. satisfying the **Occam's Razor** principle [12]?

In this paper, we make the *first* attempt towards addressing this challenge by proposing a practically **Easy Transfer Learning (EasyTL)** approach. EasyTL is able to perform knowledge transfer across domains *without* the need for model selection and hyperparameter tuning (Table 1). By exploiting *intra-domain stuctures*, EasyTL learns both *non-parametric* transfer features by *intra-domain alignment* and transfer classifier by *intra-domain programming*. Thus, it is able to avoid negative transfer [1]. Furthermore, EasyTL can also *increase* the performance of existing TL methods by serving as their final classifier via intra-domain programming.

We conduct extensive experiments on public TL datasets. Both visual domain adaptation, cross-domain sentiment analysis, and multi-source/target TL experiments demonstrated significant superiority of EasyTL in classification accuracy and computational efficiency over state-of-the-art traditional and deep TL methods.

In short, EasyTL has the following characteristics:

- *Easy:* EasyTL is extremely easy to implement and use. Therefore, it eliminates the need for model selection or hyperparameter tuning in transfer learning.
- *Accurate:* EasyTL produces competitive results in several popular TL tasks compared to state-of-the-art traditional and deep methods.
- *Efficient:* EasyTL is significantly more efficient than other methods. This makes EasyTL more suitable for resource-constrained devices such as wearables.
- *Extensible:* EasyTL can increase the performance of existing TL methods by replacing their classifier with intra-domain programming.

2. RELATED WORK

EasyTL significantly differs from existing work in the following three aspects:

Transfer learning. Existing TL methods can be summarized into two main categories: (a) *instance reweighting*, which reuses samples from the source domain according to some weighting technique; and (b) *feature transformation*, which performs subspace learning or distribution adaptation [2, 3, 13, 14, 15]. Unfortunately, these methods are all *parametric* approaches. They depend on extensive hyperparameter tuning through cross-validation for the feature transformation [13, 2], or the prediction model [6, 14], or both [15, 3]. Most methods require multiple iterations of training [15, 2]. A recent Learning to Transfer (L2T) framework [16] is similar to EasyTL in spirit, but L2T is still based on model iteration and parameter tuning. Deep TL methods [9, 17, 18, 8] require heavy hyperparameter tuning. In TL, *cross-validation* is often not available since there are almost no labeled data in the target domain [1]. In contrast, EasyTL is a non-parametric TL approach that directly learns from *intra-domain structures*, which requires no model selection and hyperparameter tuning and much more efficient than existing methods.

Non-parametric learning. Nearest-neighbor (NN) classifier is the most common non-parametric method. However, NN computes the distance between each sample in two domains, which is more likely to be influenced by domain shift. A recent Naive Bayes NN (NBNN) classifier is used for domain adaptation [19], which still requires hyperparameter tuning and iterations. Nearest Centroid (NC) classifier is based on the distance between each target sample to the class center of the source domain. The biggest difference is that EasyTL uses a linear programming to get the softmax probabilities (float weights), while NC is basically 0/1 weights. This means that EasyTL could not only consider the relationship between sample and center, but also the relations of other samples. We believe this property is important in real applications. Open set DA [20] has similar idea with EasyTL, while it solves a binary programming problem and requires other classifiers.

Automated machine learning. Recent years have witnessed the advance of Automated Machine Learning (AutoML) [11]. AutoML produces results for machine learning tasks without human intervention such as model selection and parameter tuning. However, none of the existing AutoML frameworks can handle TL tasks since these methods typically assume the training and test data are in the *same* distribution. A very recent work similar to EasyTL is AutoDIAL [21], which includes automatic domain alignment layers designed for deep networks. However, AutoDIAL still requires many parameters in the neural network to be tuned.

3. EASY TRANSFER LEARNING

We follow the most common TL settings in existing work [3, 2, 5]. We are given a labeled source domain $\Omega_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ and an unlabeled target domain $\Omega_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$. The feature space $\mathcal{X}_s = \mathcal{X}_t$, label space $\mathcal{Y}_s = \mathcal{Y}_t$, but marginal distribution $P_s(\mathbf{x}_s) \neq P_t(\mathbf{x}_t)$ with conditional distribution $Q_s(y_s|\mathbf{x}_s) \neq Q_t(y_t|\mathbf{x}_t)$. The goal is to predict the labels $\mathbf{y}_t \in \mathcal{Y}_t$ for the target domain.

3.1. Motivation

It is challenging to design a TL method that requires no model selection and hyperparameter tuning. On one hand, a simple NN (Nearest Neighbor) classifier may suffice, while NN suffers in handling the distribution divergence between domains. On the other hand, existing TL methods such as GFK [6] and BDA [15] could reduce the distribution divergence, while they require tuning a lot of hyperparameters. Combining both of them without considering domain structures may easily re-

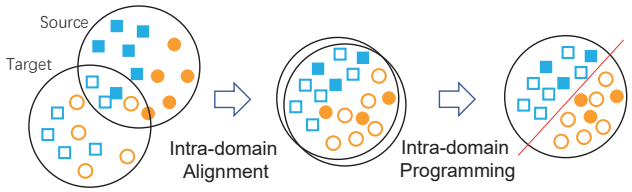


Fig. 2. The procedure of EasyTL. The colored boxes and circles denote samples from the source and target domains. The red line denotes the classifier.

sult in *negative transfer* [1], which dramatically hurts the performance of TL. Currently, there is no such effort.

In this work, we propose a practically *Easy Transfer Learning (EasyTL)* approach to learn non-parametric transfer features and classifier with competitive performance to existing *heavy* methods. In short, to be consistent with *Occam's Razor* principle [12]. In light of the recent advance in representation learning [22], incorporating knowledge of locality can greatly improve the representations quality. Therefore, Instead of learning sample-wise distance, EasyTL focuses on exploiting the *intra-domain structure*. The main part of EasyTL is a novel non-parametric *Intra-domain programming* classifier, while remains open for adopting existing methods for *Intra-domain alignment*. Intra-domain programming is able to learn discriminative transfer information from domains. Fig. 2 shows the procedure of EasyTL. In the following sections, we first introduce the proposed intra-domain programming. Then, we show how to adapt a non-parametric feature learning method for intra-domain alignment.

3.2. Intra-domain Programming

Intra-domain programming directly learns a transfer classifier for TL problem and provides reliable likelihood information for intra-domain alignment. We introduce the concept of the *Probability Annotation Matrix*, based on which we can build the non-parametric transfer classifier. Formally, let $c \in \{1, \cdots, C\}$ denote the class label, a matrix $\mathbf{M} \in \mathbb{R}^{C \times n_t}$ with its element $0 \leq M_{cj} \leq 1$ is a probability annotation matrix. Here, the entry value M_{cj} of $\mathbf{M}$ indicates the annotation probability of $\mathbf{x}_j^t$ belonging to class c. Fig. 3 illustrates the main idea of $\mathbf{M}$. There are 4 classes and n_t target samples with example activation values. Similar to the widely-adopted softmax classifier in neural networks, the highest probability value for $\mathbf{x}_1^t$ is 0.4, which indicates that it belongs to C_4. The same goes for $\mathbf{x}_2^t$ and $\mathbf{x}_3^t$, etc. The real probability annotation values are to be learned by our proposed approach.

Instead of learning $\mathbf{y}^t$ directly, the algorithm focuses on learning the probability annotation matrix $\mathbf{M}$. In this way, the cost function can be formalized as:

$$\mathcal{J} = \sum_{j}^{n_t} \sum_{c}^{C} D_{cj} M_{cj}, \tag{1}$$

	$\mathbf{x}_1^t$	$\mathbf{x}_2^t$	$\mathbf{x}_3^t$	.....	$\mathbf{x}_{n_t}^t$
C_1	0.1	0.2	**0.5**		.....
C_2	0.2	**0.4**	0.1		.....
C_3	0.3	0.2	0.2		.....
C_4	**0.4**	0.2	0.2		.....

Fig. 3. An example of the probability annotation matrix.

where the distance value D_{cj} is an entry in a distance matrix $\mathbf{D}$. D_{cj} denotes the distance between $\mathbf{x}_j^t$ and the c-th class center of the source domain $\Omega_s^{(c)}$.

We denote $\mathbf{h}_c$ as the c-th class center of $\Omega_s^{(c)}$. Then, D_{cj} can be computed by the Euclidean distance:

$$D_{cj} = ||\mathbf{x}_j^t - \mathbf{h}_c||^2. \tag{2}$$

The class center $\mathbf{h}_c$ on the labeled source domain can be calculated as:

$$\mathbf{h}_c = \frac{1}{|\Omega_s^{(c)}|} \sum_{i}^{n_s} \mathbf{x}_i^s \cdot \mathbb{I}(y_i^s = c), \tag{3}$$

where $\mathbb{I}(\cdot)$ is an indicator function which evaluates to 1 if the condition is true, and 0 otherwise.

Consider the constraints to minimize the cost function in Eq. (1). Firstly, note that the value of M_{cj} is a probability measuring the confidence of $\mathbf{x}_j^t$ belonging to class c, such that the sum of the probability of one particular sample $\mathbf{x}_j^t$ belonging to all existing classes is 1. This is ensured by the following constraint:

$$\sum_{c}^{C} M_{cj} = 1, \forall j \in \{1, \cdots, n_t\}. \tag{4}$$

Secondly, since Ω_s and Ω_t have the same label space (i.e. $\mathcal{Y}_s = \mathcal{Y}_t$), there must be at least one sample for any given class c. This is ensured by the following constraint:

$$M_{cj} = \arg\max_{r} M_{rj}, r \in \{1, \cdots, C\}, \forall c \in \{1, \cdots, C\}, \exists j. \tag{5}$$

In fact, the ideal state of M_{cj} should be a binary value (0 or 1), i.e. $M_{cj} = 1$ iff $\mathbf{x}_j$ belongs to class c, otherwise $M_{cj} = 0$. Therefore, we use the following formulation to replace Eq. (5) in the computation without affecting the results:

$$\sum_{j}^{n_t} M_{cj} \geq 1, \forall c \in \{1, \cdots, C\}. \tag{6}$$

Learning objective. Combining the cost function in Eq. (1) and the constraints in Eq. (4) and Eq. (6), the final

learning objective becomes:

$$\min \quad \mathcal{J} = \sum_{j}^{n_t} \sum_{c}^{C} D_{cj} M_{cj}. \\ s.t. \quad \begin{cases} 0 \leq M_{cj} \leq 1 \\ \sum_{c}^{C} M_{cj} = 1, \forall j \in \{1, \cdots, n_t\} \\ \sum_{j}^{n_t} M_{cj} \geq 1, \forall c \in \{1, \cdots, C\} \end{cases} \tag{7}$$

Solving Eq. (7) requires us to solve a linear programming (LP) problem. There are existing solution packages such as PuLP [1] for solving the above linear programming problem efficiently. Then, the probability annotation matrix can be obtained. Eventually, the label of $\mathbf{x}_j^t$ is given by the softmax function:

$$y_j^t = \arg\max_r \ \frac{M_{rj}}{\sum_c^C M_{cj}} \quad \text{for } r \in \{1, \cdots, C\}. \tag{8}$$

It is noticeable that this classifier does **not** involve any parameters that require further tuning. This is significantly different from existing well-established classifiers such as SVM that need to tune numerous hyperparameters. Additionally, intra-domain programming can be used alone for TL problems.

3.3. Intra-domain Alignment

Intra-domain alignment serves as the transfer feature learning methods for EasyTL. Although EasyTL can be used directly in TL with good performance, it can also be combined with transfer feature learning to eliminate feature distortions in the original space. On the other hand, existing transfer feature learning methods can also be extended using *intra-domain programming* to enhance their performance.

Recall our inspiration that learning the structure of locality will great help learn useful representations [22]. Therefore, if we perform *personalized* transfer feature learning within each subspace, we could further reduce the domain divergence. For simplicity and efficiency, we only perform feature learning from the source domain to the subspace of the target domain.

Inspired by the non-parametric feature learning method CORAL [14], the intra-programming process of EasyTL can be formulated as:

$$\mathbf{z}^r = \begin{cases} \mathbf{x}^r \cdot (cov(\mathbf{x}^s) + \mathbf{E}_s)^{-\frac{1}{2}} \cdot (cov(\mathbf{x}^t) + \mathbf{E}_t)^{\frac{1}{2}} & \text{if } r = s \\ \mathbf{x}^r & \text{if } r = t \end{cases} \tag{9}$$

where $cov(\cdot)$ is the covariance matrix. $\mathbf{E}_s$ and $\mathbf{E}_t$ are identity matrices with equal sizes to Ω_s and Ω_t, respectively. We can treat this step as a *re-coloring* process of each subspace [14]. Eq. (9) aligns the two distributions by re-coloring whitened source features with the covariance of target distributions.

[1] https://pypi.python.org/pypi/PuLP/1.1

Algorithm 1 EasyTL: Easy Transfer Learning

Input: Feature matrix $\mathbf{x}_s, \mathbf{x}_t$ for Ω_s and Ω_t, respectively; and label vector $\mathbf{y}_s$ for Ω_s

Output: Predicted label vector $\mathbf{y}_t$ for target domain.

1: (Optional) Perform intra-domain alignment according to Eq. (9)
2: Solve Eq. (7) to obtain the probability annotation matrix $\mathbf{M}$ and compute $\mathbf{y}_t$ using Eq. (8)
3: **return** Label vector $\mathbf{y}_t$ for Ω_t

Remark: Other than CORAL, EasyTL can also choose other popular methods for feature learning such as BDA [15] and GFK [6]. We choose CORAL for its computational efficiency and that it contains no other parameters to tune.

3.4. Computational Complexity

We use the big-O notation to analyze the complexity of EasyTL. Intra-domain alignment takes at most $O(Cn_s^3)$. The complexity of intra-domain programming is $O(n_t^3 C^3)$ given that it is a linear programming problem (Eq. (7)) [23]. EasyTL can be made more efficient with the low-rank representations and other fast computing algorithms.

As an intuitive comparison of classifiers, we compare intra-domain programming with the well-established SVM. A *single* round of SVM training and prediction takes $O((n_s + n_t)^3)$ computations [24]. This implies that the time complexity of intra-domain programming is comparable to a single round of SVM. However, SVM still needs a couple of rounds for hyperparameter tuning (e.g. *kerneltype*, *constraints*) before getting optimal performance. Therefore, EasyTL is theoretically more efficient than SVM. We will experimentally show the efficiency of EasyTL in the next sections.

4. EXPERIMENTAL EVALUATION

4.1. Experimental Setup

We use four popular TL datasets: 1) Amazon Review [26], 2) Office-Caltech, 3) Image-CLEF DA [9], and 4) Office-Home [7]. **Amazon Review** is a cross-domain sentiment analysis dataset that contains positive and negative reviews of four kinds of products: Kitchen appliance (K), DVDs (D), Electronics (E), and Books (B). **Office-Caltech** dataset contains 10 common classes of images in Amazon (A), DSLR (D), Webcam (W), and Caltech (C). **Image-CLEF DA** dataset contains 12 categories of images where each category has 600 images. There are 3 domains: Caltech (C), ImageNet (I), and Pascal (P). **Office-Home** dataset contains 15,500 images of 65 categories from 4 domains: Art (Ar), Clipart (Cl), Product (Pr), and Real-world (Rw). Within each dataset, any two domains can be source and target domains to construct transfer learning tasks.

Table 2. Accuracy (%) on Image-CLEF DA and Office-Home datasets using ResNet features

ID	Task	ResNet	1NN	SVM	TCA	GFK	BDA	CORAL	DANN	JAN	CDAN	EasyTL(c)	EasyTL
1	C → I	78.0	83.5	86.0	89.3	86.3	90.8	83.0	87.0	89.5	91.2	85.5	**91.5**
2	C → P	65.5	71.3	73.2	74.5	73.3	73.7	71.5	74.3	74.2	77.2	72.0	**77.7**
3	I → C	91.5	89.0	91.2	93.2	93.0	94.0	88.7	96.2	94.7	96.7	93.3	**96.0**
4	I → P	74.8	74.8	76.8	77.5	75.5	75.3	73.7	75.0	76.8	78.3	78.5	**78.7**
5	P → C	91.2	76.2	85.8	83.7	82.3	83.5	72.0	91.5	91.7	93.7	91.0	**95.0**
6	P → I	83.9	74.0	80.2	80.8	78.0	77.8	71.3	86.0	88.0	**91.2**	89.5	90.3
-	AVG	80.7	78.1	82.2	83.2	81.4	82.5	76.7	85.0	85.8	88.1	85.0	**88.2**
7	Ar → CI	34.9	45.3	45.3	38.3	38.9	38.9	42.2	45.6	45.9	46.6	51.6	**52.8**
8	Ar → Pr	50.0	60.1	65.4	58.7	57.1	54.8	59.1	59.3	61.2	65.9	68.1	**72.1**
9	Ar → Rw	58.0	65.8	73.1	61.7	60.1	58.2	64.9	70.1	68.9	73.4	74.2	**75.9**
10	CI → Ar	37.4	45.7	43.6	39.3	38.7	36.2	46.4	47.0	50.4	55.7	53.1	**55.0**
11	CI → Pr	41.9	57.0	57.3	52.4	53.1	53.1	56.3	58.5	59.7	62.7	62.9	**65.9**
12	CI → Rw	46.2	58.7	60.2	56.0	55.5	50.2	58.3	60.9	61.0	64.2	65.3	**67.6**
13	Pr → Ar	38.5	48.1	46.8	42.6	42.2	42.1	45.4	46.1	45.8	51.8	52.8	**54.4**
14	Pr → CI	31.2	42.9	39.1	37.5	37.6	38.2	41.2	43.7	43.4	**49.1**	45.8	46.9
15	Pr → Rw	60.4	68.9	69.2	64.1	64.6	63.1	68.5	68.5	70.3	74.5	73.5	**74.7**
16	Rw → Ar	53.9	60.8	61.1	52.6	53.8	50.2	60.1	63.2	63.9	**68.2**	62.2	63.8
17	Rw → CI	41.2	48.3	45.6	41.7	42.3	44.0	48.2	51.8	52.4	**56.9**	50.2	52.3
18	Rw → Pr	59.9	74.7	75.9	70.5	70.6	68.2	73.1	76.8	76.8	**80.7**	76.0	78.0
-	AVG	46.1	56.4	56.9	51.3	51.2	49.8	55.3	57.6	58.3	62.8	61.3	**63.3**
-	Average rank [25]	12	7	6	9	10	11	8	5	4	2	3	**1**

Table 3. Accuracy (%) on Amazon Review dataset

Method	1NN	TCA	GFK	SA	BDA	CORAL	JGSA	EasyTL(c)	EasyTL
B → D	49.6	63.6	66.4	67.0	64.2	71.6	66.6	78.4	**79.8**
B → E	49.8	60.9	65.5	70.8	62.1	65.1	75.0	77.5	**79.7**
B → K	50.3	64.2	69.2	72.2	65.4	67.3	72.1	79.2	**80.9**
D → B	53.3	63.3	66.3	67.5	62.4	70.1	55.5	79.5	**79.9**
D → E	51.0	64.2	63.7	67.1	66.3	65.6	67.3	77.4	**80.8**
D → K	53.1	69.1	67.7	69.4	68.9	67.1	65.6	80.4	**82.0**
E → B	50.8	59.5	62.4	61.4	59.2	67.1	51.6	73.0	**75.0**
E → D	50.9	62.1	63.4	64.9	61.6	66.2	50.8	73.1	**75.3**
E → K	51.2	74.8	73.8	70.4	74.7	77.6	55.0	84.6	**84.9**
K → B	52.2	64.1	65.5	64.4	62.7	68.2	58.3	75.2	**76.5**
K → D	51.2	65.4	65.0	64.6	64.3	68.9	56.4	73.8	**76.3**
K → E	52.3	74.5	73.0	68.2	74.0	75.4	51.7	82.0	**82.5**
AVG	51.3	65.5	66.8	67.3	65.5	69.1	60.5	77.8	**79.5**

State-of-the-art traditional and deep comparison methods: **NN** (Nearest Neighbor), **SVM** with linear kernel [14], **TCA** (Transfer Component Analysis) [13], **GFK** (Geodesic Flow Kernel) [6], **SA** (Subspace Alignment) [27], **CORAL** (CORrelation ALignment) [14], **BDA** (Balanced Distribution Adaptation) [15], **JGSA** (Joint Geometrical and Statistical Alignment) [5], **D-GFK** [2], **ResNet50** [10], **DANN** (Domain-adversarial Neural Networks) [8], **JAN** (Joint Adaptation Networks) [17], and **CDAN** (Conditional Adversarial Adaptation Networks) [9]. The source code of EasyTL is available at `http://transferlearning.xyz/code/traditional/EasyTL`.

Specifically, we use *EasyTL(c)* to denote intra-domain programming since it can serve as a TL method alone, while *EasyTL* denotes the full method. By following the standard protocol in [14], we adopt linear SVM for traditional TL methods. For the amazon dataset, we use the 400-dimensional features provided by [26]. For image datasets, we use the 2048-dimensional ResNet50 [10] finetuned features provided by [28]. For Office-Caltech datasets, we adopt the SURF features [2]. Deep TL methods are only used in image datasets and their results are obtained from existing work. Classification accuracy is acting as the evaluation metric [13, 6, 14].

Table 4. Average rank, parameter, and running time

Method	TCA	GFK	CORAL	DANN	CDAN	EasyTL(c)	EasyTL
Avg rank	9	10	8	4	2	3	**1**
Parameter	d, λ, C, γ	d, C, γ	C, γ	λ	λ	**None**	**None**
Time (s)	119	127.4	126.5	>1000	>1000	**23.8**	59.6

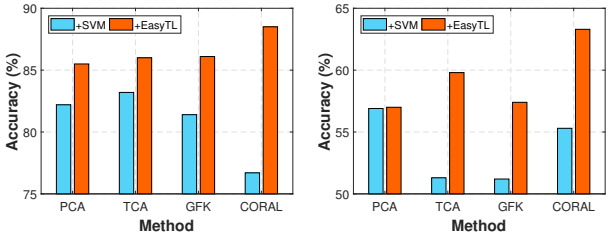
4.2. Results: Accuracy and Efficiency

The results on Amazon Review is shown in Table 3. Table 2 shows the results on ImageCLEF DA (IDs 1∼6) and Office-Home (IDs 7∼18). The results on Office-Caltech are shown in Table 5. We also report the average rank [25], parameter and running time (train+test) of several methods in Table 4. The results demonstrate that EasyTL outperforms all comparison methods in both sentiment and image data. EasyTL(c) can also achieve competitive performance than most methods. Note that except 1NN and EasyTL, other methods all require hyperparameter tuning either in feature or classifier learning. These results clearly indicate the superiority of EasyTL in accuracy and efficiency.

There are more insightful findings. 1) On the larger and more challenging Office-Home dataset, the performance of EasyTL(c) is only slightly worse than CDAN, while it outperforms all other comparison methods. Compared to deep TL methods (DANN, JAN, and CDAN), although EasyTL takes finetuned features as inputs, it only requires *one* distinct finetuning process while deep methods have to run the network multiple times to get optimal parameters. 2) Moreover, in real applications, it is rather difficult and almost *impossible* to search parameters since there are often little or none labeled data in the target domain. The results seems that adversarial learning provides little contribution on domain adaptation when compared to the power of ResNet in representation learning provided that it is rather difficult to train an adversarial network. 3) EasyTL tends to perform consistently well on rather balanced datasets (Amazon Review and Image-

Table 5. Accuracy (%) on Office-Caltech dataset using SURF features

Method	NN	AutoML	SVM	TCA	SA	CORAL	GFK	BDA	JGSA	D-GFK	EasyTL(c)	EasyTL
C → A	23.7	37.4	53.1	45.8	39.0	47.2	46.0	44.9	51.5	**54.4**	50.1	52.6
C → W	25.8	26.6	41.7	31.2	36.8	39.2	37.0	38.6	45.4	46.4	49.5	**53.9**
C → D	25.5	34.8	47.8	**34.4**	39.6	40.7	40.8	47.8	45.9	49.7	48.4	**51.6**
A → C	25.0	38.2	41.7	42.4	35.3	40.3	40.7	40.8	41.5	**45.7**	43.0	42.3
A → W	29.8	31.3	31.9	36.3	38.6	38.7	37.0	39.3	**45.8**	41.7	40.7	43.1
A → D	25.5	30.8	44.6	33.8	37.6	38.3	40.1	43.3	47.1	46.5	38.9	**48.4**
W → C	19.9	31.4	21.2	29.4	32.3	34.6	24.8	28.9	33.2	35.1	29.7	**35.4**
W → A	23.0	34.5	27.6	28.9	37.4	37.8	27.6	33.0	**39.9**	38.6	35.2	38.2
W → D	59.2	78.2	78.3	89.2	80.3	84.9	85.4	**91.7**	90.5	90.5	77.1	79.6
D → C	26.3	29.2	26.4	30.7	32.4	34.2	29.3	32.5	29.9	32.1	31.2	**36.1**
D → A	28.5	32.3	26.2	31.0	38.0	38.1	28.7	33.1	38.0	38.1	31.9	**38.3**
D → W	63.4	67.8	52.5	86.1	83.6	85.9	80.3	**91.9**	**91.9**	84.4	69.5	86.1
AVG	31.4	39.4	41.1	43.3	44.2	46.7	43.1	47.2	50.0	50.3	45.4	**50.5**



(a) Image-CLEF DA (b) Office-Home

Fig. 4. Extending existing methods with EasyTL

CLEF DA). While on the unbalanced Office-Caltech dataset, the performance of EasyTL is limited. The research of improving EasyTL for unbalanced datasets remains as future research. 4) The results of EasyTL imply that the "2-stage" procedure (finetune+EasyTL) is better than "1-stage" methods (deep TL) even if they are end-to-end. All methods have their advantages and disadvantages.

4.3. Evaluation of Extensibility

We evaluate the extensibility of EasyTL by extending existing transfer feature learning methods with the classifier of EasyTL. We use PCA, TCA [13], GFK [6], and CORAL [14] for intra-domain alignment and compare the performance of +SVM and +EasyTL in Fig. 4. The results of each dataset show that: (1) By using feature learning methods other than CORAL, EasyTL can still achieve comparable performance to existing approaches. (2) More importantly, EasyTL does **not** rely on any particular feature learning methods to achieve good performance. (3) EasyTL could *increase* the performances of existing transfer learning methods. These results clearly imply the extensibility of EasyTL in transfer learning.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we propose EasyTL as the first non-parametric transfer learning approach by exploiting intra-domain structures features to learn non-parametric transfer features and classifiers. Thus, it requires no model selection and hyperparameter tuning. EasyTL is easy, accurate, efficient, and extensible, and can be applied directly in the resource-constrained devices such as wearables. Experiments demonstrate the superiority of EasyTL in accuracy and efficiency over several popular traditional and deep TL methods. EasyTL can also increase the performance of other TL methods.

EasyTL opens up some exciting new research opportunities in transfer learning:

Practical Transfer Learning. Instead of pursuing good performance in accuracy, we do hope that EasyTL could open up a new door in research on practical transfer learning that focuses more on the actual usage of the methods: parameter tuning and model validation.

Online Transfer Learning. Despite the accuracy and efficiency achieved by EasyTL, EasyTL can only handle with offline data currently. It would be beneficial to apply EasyTL to online situations where data are coming in stream. In this way, EasyTL can be applied to real-time devices.

Automated Domain Selection. Currently, EasyTL focuses on the problem of *how to transfer*. In real applications with data from different domains, another important problem is *what to transfer*. Some existing work [29] focused on this issue, but they also require heavy parameter tuning. We will further explore this part by proposing automated transfer methods based on EasyTL.

Multi-label Transfer Learning. EasyTL can be used in multi-label [30] transfer learning by changing the value of probability annotation matrix. This makes EasyTL suitable for multi-label tasks. Existing multi-label transfer methods have to modify their classifiers. EasyTL can be more advantageous for these applications.

6. ACKNOWLEDGMENTS

This work is supported by National Key R & D Program of China (2016YFB1001200), NSFC (61572471, 61702520), Hong Kong CERG projects (16209715, 16244616), and Nanyang Technological University, Nanyang Assistant Professorship (NAP).

7. REFERENCES

[1] Sinno Jialin Pan and Qiang Yang, "A survey on transfer learning," *IEEE TKDE*, vol. 22, no. 10, pp. 1345–1359, 2010.

[2] Jianre Wei, Jian Liang, Ran He, and Jinfeng Yang, "Learning discriminative geodesic flow kernel for unsupervised domain adaptation," in *ICME*, 2018, pp. 1–6.

[3] Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S. Yu, "Visual domain adaptation with manifold embedded distribution alignment," in *ACM Multimedia (ACM MM)*, 2018, pp. 402–410.

[4] Jindong Wang, Yiqiang Chen, Lisha Hu, Xiaohui Peng, and Philip S Yu, "Stratified transfer learning for cross-domain activity recognition," in *IEEE PerCom*, 2018.

[5] Jing Zhang, Wanqing Li, and Philip Ogunbona, "Joint geometrical and statistical alignment for visual domain adaptation," in *CVPR*, 2017.

[6] Boqing Gong, Yuan Shi, et al., "Geodesic flow kernel for unsupervised domain adaptation," in *CVPR*, 2012, pp. 2066–2073.

[7] Hemanth Venkateswara, Jose Eusebio, et al., "Deep hashing network for unsupervised domain adaptation," in *CVPR*, 2017, pp. 5018–5027.

[8] Yaroslav Ganin and Victor Lempitsky, "Unsupervised domain adaptation by backpropagation," in *ICML*, 2015, pp. 1180–1189.

[9] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan, "Conditional adversarial domain adaptation," in *NeurIPS*, 2018, pp. 1647–1657.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[11] Yao Quanming et al., "Taking human out of learning applications: A survey on automated machine learning," *arXiv preprint:1810.13306*, 2018.

[12] Carl Edward Rasmussen and Zoubin Ghahramani, "Occam's razor," in *NeurIPS*, 2001, pp. 294–300.

[13] Sinno Jialin Pan, Qiang Yang, et al., "Domain adaptation via transfer component analysis," *IEEE TNN*, vol. 22, no. 2, pp. 199–210, 2011.

[14] Baochen Sun, Jiashi Feng, and Kate Saenko, "Return of frustratingly easy domain adaptation.," in *AAAI*, 2016, vol. 6, p. 8.

[15] Jindong Wang, Yiqiang Chen, Shuji Hao, et al., "Balanced distribution adaptation for transfer learning," in *ICDM*, 2017, pp. 1129–1134.

[16] Ying Wei, Yu Zhang, and Qiang Yang, "Transfer learning via learning to transfer," in *ICML*, 2018.

[17] Mingsheng Long, Jianmin Wang, et al., "Deep transfer learning with joint adaptation networks," in *ICML*, 2017, pp. 2208–2217.

[18] Eric Tzeng, Kate Saenko, Trevor Darrell, et al., "Adversarial discriminative domain adaptation," in *CVPR*, 2017, pp. 2962–2971.

[19] Tatiana Tommasi and Barbara Caputo, "Frustratingly easy nbnn domain adaptation," in *ICCV*, 2013, pp. 897–904.

[20] Pau Panareda Busto and Juergen Gall, "Open set domain adaptation," in *ICCV*, 2017, pp. 754–763.

[21] Fabio Maria Carlucci, Lorenzo Porzi, et al., "Autodial: Automatic domain alignment layers," in *ICCV*, 2017, pp. 5067–5075.

[22] R Devon Hjelm, Alex Fedorov, et al., "Learning deep representations by mutual information estimation and maximization," in *ICLR*, 2019.

[23] Nimrod Megiddo, *On the complexity of linear programming*, 1986.

[24] Abdiansah Abdiansah and Retantyo Wardoyo, "Time complexity analysis of support vector machines in libsvm," *J. Comput. Appl.*, vol. 128, no. 3, pp. 28–34, 2015.

[25] Janez Demšar, "Statistical comparisons of classifiers over multiple data sets," *JMLR*, vol. 7, no. Jan, pp. 1–30, 2006.

[26] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha, "Marginalized denoising autoencoders for domain adaptation," in *ICML*, 2012.

[27] Basura Fernando et al., "Unsupervised visual domain adaptation using subspace alignment," in *ICCV*, 2013, pp. 2960–2967.

[28] Jindong Wang et al., "Everything about transfer learning and domain adapation," `http://transferlearning.xyz`, 2018.

[29] Evan Wei Xiang, Sinno Jialin Pan, Weike Pan, Jian Su, and Qiang Yang, "Source-selection-free transfer learning," in *IJCAI*, 2011, vol. 22, p. 2355.

[30] Sheng-Jun Huang, Wei Gao, and Zhi-Hua Zhou, "Fast multi-instance multi-label learning," *IEEE TPAMI*, 2018.