A Hybrid Evolutionary Computation Approach to Inducing Transfer Classifiers for Domain Adaptation

Bach Hoai Nguyen, *Member, IEEE*, Bing Xue, *Member, IEEE*, Peter Andreae, and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Domain adaptation utilizes learned knowledge from an existing domain (source domain) to improve the classification performance of another related, but not identical, domain (target domain). Most existing domain adaptation methods firstly perform domain alignment, then apply standard classification algorithms. Transfer classifier induction is an emerging domain adaptation approach that incorporates the domain alignment into the process of building an adaptive classifier instead of using a standard classifier. Although transfer classifier induction approaches have achieved promising performance, they are mainly gradient-based approaches which can be trapped at local optima. In this paper, we propose a transfer classifier induction algorithm based on evolutionary computation to address the above limitation. Specifically, a novel representation of the transfer classifier is proposed which has much lower dimensionality than the standard representation in existing transfer classifier induction approaches. We also propose a hybrid process to optimize two essential objectives in domain adaptation: the manifold consistency and the domain difference. Particularly, the manifold consistency is used in the main fitness function of the evolutionary search to preserve the intrinsic manifold structure of the data. The domain difference is reduced via a gradient-based local search applied to the top individuals generated by the evolutionary search. The experimental results show that the proposed algorithm can achieve better performance than seven state-of-the-art traditional domain adaptation algorithms and four state-of-the-art deep domain adaptation algorithms.**

***Index Terms*—Domain adaptation, transfer learning, classification, evolutionary computation**

I. Introduction

Classification — one of the most important tasks in machine learning [1], [2], [3] — typically requires labeled data to learn an effective classifier. In many real-world scenarios, it is expensive to manually label instances from a new domain. A possible solution is to train a classification algorithm using labeled data from a related (source) domain. The obtained classifier can be used to classify the unlabeled instances in the new (target) domain. However, the two domains may have different feature spaces and/or different data distributions. Such differences cause a significantly degraded classification performance on the target domain since most standard classification algorithms assume that the training and test sets should be from the same domain. The situation promotes the need for transfer learning [4] which aims to reduce the domain differences. Domain adaptation [4] is a particular case of transfer learning, where the two domains share the same feature space. Domain adaptation focuses on reducing the data distribution difference. Some techniques for domain adaptation require labeled data in the target domain to guide the domain adaptation process. A more difficult problem is unsupervised domain adaptation where the target domain has no labels. This paper address the unsupervised domain adaptation problem.

The two most common domain adaptation approaches are instance-based and feature-based methods [4]. Instance-based methods attempt to weight instances from both domains such that the distribution difference is reduced. For example, TrAdaBoost [5] trains a classifier based on both labelled source instances and a few labelled target instances. If any source instance is wrongly classified by the learned classifier, the source instance is considered to have a different distribution in comparison with the target instances and its weight is reduced. After a certain number of iterations, the source instances, which form similar distributions to the target instances, will have larger weights. Such source instances can be used to improve the classification performance on the target domain. Feature-based methods, on the other hand, attempt to learn a feature transformation where the transformed source and target data are more similar, which is also known as domain alignment [6], [7]. Both instance-based and feature-based methods are two-step algorithms, where the first step is to reduce the domain difference and the second step is to apply a standard classification algorithm, such as k-nearest neighbors (KNN). Such a mechanism does not guarantee that the standard classification can cope well with the weighted or transformed data. Recently, transfer classifier induction has been proposed [8] to directly learn an adaptive classifier rather than using a standard classification algorithm on the transformed data. The adaptive classifier embeds the domain alignment into its learning process. Since the transfer classifier induction method takes the classifier into account during the domain alignment, it can achieve superior performance compared to the instance-based and feature-based approaches [8], [9]. Therefore, this paper also aims to build an adaptive classifier for domain adaptation.

Although transfer classifier induction approaches achieve promising results, most of them are based on the use of gradients, which makes it easier for them to be trapped in local optima. Evolutionary computation (EC) is a family of population-based optimization approaches that are well-known because of their potential global search ability. In [10], we

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1509 and VUW1615, the Science for Technological Innovation Challenge (SfTI) fund under grant E3603/2903, and the University Research Fund at Victoria University of Wellington grant number 216378/3764 and 223805/3986.

The authors are with the Evolutionary Computation Research Group, Victoria University of Wellington, Wellington, New Zealand (e-mail: Hoai.Bach.Nguyen@ecs.vuw.ac.nz).

proposed the *first* EC-based approach for transfer classifier induction, called P-MEDA. In that algorithm, each candidate solution (or population member) represents a classifier which is essentially a matrix of continuous numbers. This representation has several limitations which lead to the restricted performance of P-MEDA. Firstly, since the matrix scales according to the total number of instances in both domains and the number of class labels, it is a high-dimensional optimization problem. Secondly, the matrix elements are unbounded, so standard genetic operators, such as crossover and mutation, are less effective. Although P-MEDA is a population-based approach, the main step to evolve the candidate solution is still gradient-based. The population members communicate only when a candidate solution needs to be re-initialized. The communication is also limited, mainly via an archive set which records all the promising classifiers discovered so far. Thirdly, each evaluation of P-MEDA requires computing many matrix operations, which is computationally expensive, especially when there is a large number of instances.

This work extends our previous work [10] to address the three limitations. The contributions of this work are as follows:

- Firstly, we propose a new discrete representation which has much lower dimensionality than the continuous representation in existing transfer classifier induction algorithms. The proposed representation can directly optimize the class labels for target instances.
- Secondly, we propose a hybrid approach that combines the evolutionary search and the gradient search. Note that the proposed hybrid approach is significantly different from the hybrid approach in P-MEDA [10]. Although P-MEDA uses some ideas from the evolutionary search, it still relies mainly on the gradient search due to its high-dimensional representation. In contrast, the proposed algorithm uses the evolutionary search as the main component to reduce the manifold inconsistency, and the gradient search plays the role of a local search to further refine the candidate solutions to reduce the domain discrepancy. Such difference is because standard EC genetic operators can be naturally applied to the proposed discrete representation.
- Thirdly, we propose a more efficient fitness function which uses fewer matrix operations in each evaluation with an expectation of improving efficiency.

The proposed algorithm is validated by comparing it with three well-known classification algorithms, seven state-of-the-art traditional domain adaptation approaches, and four state-of-the-art deep domain adaptation approaches. The comparison is performed on 30 real-world domain adaptation cases.

The rest of the paper is organized as follows: Section II provides a discussion about existing domain adaptation approaches, especially the transfer classifier induction category. Section III introduces the proposed algorithm. Section IV summarizes the experimental studies. The conclusions and future work are discussed in Section V.

II. Related work

This section firstly introduces basic concepts in transfer learning and domain adaptation. Next, a subsection is dedicated to discussing transfer classifier induction which is the focus of this paper. Notations that are frequently used in this paper are summarized in Table I.

TABLE I: Notations used in this paper.

Notation	Description	Notation	Description
D_s, D_t	source/target domain	$\mathbf{X}$	data matrix
n, m	# instances in D_s, D_t	$\mathbf{Y}$	label matrix
C	# classes	$\mathbf{L}$	Laplacian matrix
$\boldsymbol{\alpha}$	class parameters	$\mathbf{E}$	domain indicator matrix
$\boldsymbol{z}$	candidate classifier	$\mathbf{M}$	MMD matrix

A. Transfer Learning and Domain Adaptation

An essential concept in transfer learning is the *domain* which consists of two components: a feature space $\mathcal{X}$ and a marginal distribution $P(\mathcal{X})$. Two domains are different if they have different feature spaces and/or marginal distributions. Given a domain $D(\mathcal{X}, P(\mathcal{X}))$, a learning task is to find a prediction function $f(\cdot)$ that maps from the feature space $\mathcal{X}$ to a label space $\mathcal{Y}$. In a classification task, $f(\cdot)$ can be represented by a conditional distribution $Q(\mathcal{Y}|\mathcal{X})$. Two tasks are different if they have different label spaces and/or conditional distributions. Given a source domain D_s and its learning task $\mathcal{T}_s$, the goal of transfer learning is to utilize the knowledge from the source domain to improve the learning performance of the learning task $\mathcal{T}_t$ on a target domain D_t. *Domain adaptation* is a specific case of transfer learning with an assumption that the two domains have the same feature space ($\mathcal{X}_s = \mathcal{X}_t = \mathcal{X}$) and the two learning tasks have the same label space ($\mathcal{Y}_s = \mathcal{Y}_t = \mathcal{Y}$). Therefore, domain adaptation aims to reduce the differences between the two marginal distributions ($P_s(\mathcal{X}) \neq P_t(\mathcal{X})$) and the two conditional distributions ($Q_s(\mathcal{Y}|\mathcal{X}) \neq Q_t(\mathcal{Y}|\mathcal{X})$). We focus on handling domain adaptation, more specifically unsupervised domain adaptation, where there is no labeled instances in the target domain. The main task of this work can be formally defined as follows:

Unsupervised domain adaptation: *Given a labeled source data with n instances, $D_s = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n}$, and an unlabeled target data with m instances, $D_t = \{\boldsymbol{x}_j\}_{j=n+1}^{n+m}$, unsupervised domain adaptation learns a classifier $f : \boldsymbol{x}_t \rightarrow y_t$ with a low classification error on D_t, where $\mathcal{X}_s = \mathcal{X}_t, \mathcal{Y}_s = \mathcal{Y}_t, P_s(\boldsymbol{x}_s) \neq P_t(\boldsymbol{x}_t)$, and $Q(y_s|\boldsymbol{x}_s) \neq Q(y_t|\boldsymbol{x}_t)$* [8].

Existing domain adaptation approaches can be divided into two main categories: deep domain adaptation and shallow domain adaptation [11]. Deep domain adaptation embeds domain alignment into deep networks to learn discriminative and domain-invariant deep feature representations, which can significantly improve the learning performance on the target domain. Existing deep domain adaptation methods can be further divided into three categories: reconstruction-based methods, discrepancy-based methods, and adversarial-based methods [12]. The reconstruction-based methods use a stack auto-encoder to learn a domain-invariant representation for the data reconstruction of the source and target data [13], [14]. The discrepancy-based methods explicitly consider the domain discrepancy by embedding an additional loss into the objective function. Commonly used discrepancy metrics include maximum mean discrepancy (MMD) [15], [16], [17] and correlation alignment (CORAL) [18], [19]. Adversarial-based methods are increasingly popular recently [20], [11], [21]. In this case, a domain discriminator, which can identify whether an instance is from the source domain or from the target domain, is added to a deep network. The goal is to learn a deep representation to confuse the domain discriminator.

Although deep domain adaptation methods achieve promising results, they are mainly applied to visual domain adaptation. In contrast, shallow or traditional domain adaptation methods are more generalized in terms of applications. The three most common shallow domain adaptation approaches are instance-based approaches, feature-based approaches, and classifier-based approaches (i.e. transfer classifier induction) [4]. The instance-based approach considers that some source instances are more relevant to the target domain than other source instances. Therefore, this approach aims to re-weight or to select source instances such that the domain discrepancy is reduced. After that, a standard classification can be trained on the re-weighted source data to classify the target data [22], [23], [24]. The feature-based approach attempts to find a feature transformation such that the discrepancy between the transformed source and target data is minimized. An early feature-based work is Transfer Component Analysis (TCA) [25] where the goal is to extract a common latent subspace such that the data variance is preserved and the distribution difference, measured by MMD, is reduced. However, TCA assumes that the two domains have the same conditional distribution. Joint Domain Adaptation [26] addresses the limitation of TCA by jointly adapting both marginal and conditional distributions of the two domains. In addition to MMD, many other discrepancy measures have been used such as Bregman divergence [27], and the Hilbert-Schmidt independence criterion (HSIC) [28], [29]. Recently, Joint Geometrical and Statistical Alignment (JSGA) [30] incorporates geometrical information, such as the variance of the target domain and the discriminative information in the source domain, to build two coupled feature transformations for the source and target domains. With an assumption that the source and target data can be represented by a low-dimensional linear space, Gopalan et al. [31] apply principle component analysis (PCA) to form a Grassmann manifold, where the two domains are represented as two points. A geodesic path between the two points reveals information about domain changes. Many subspaces are drawn from the path to find an intermediate subspace on which the domain distance is smaller. Geodesic Flow Kernel (GFK) [32] extends the idea in [31] by considering an infinite number of subspaces. Instead of building a common subspace, some other feature-based methods aim to directly map from the source feature space to the target feature space [33], [34].

In both feature-based and instance-based approaches, domain alignment is performed first, and then a standard classifier is trained on the transformed source data to classify the transformed target data. Domain alignment using feature transformation or re-weighting instances is important in domain adaptation [35], [36], but learning a standard classifier as a side-step may not guarantee the suitability between the classifier and the transformed or re-weighted data. To address the limitation, transfer classifier induction is proposed to embed the domain alignment process into the process of building an adaptive classier, which results in its superiority over the other two approaches. Therefore, we focus on transfer classifier induction in this paper. The following subsection discusses transfer classifier induction in more detail.

B. Transfer Classifier Induction

A general frame work of transfer classifier induction (ARTL) was formally introduced by Long et al. [8], where the classifier model is $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$: $\mathbf{w}$ contains the classifier parameters, and ϕ is a feature mapping function that projects an original feature vector to a Hilbert space. Formally, a domain invariant classifier f can be learned by minimizing the following objective function:

$$Objective = (\sum_{i=1}^{n} l(f(\mathbf{x}_i), y_i) + \eta||f||_K^2) + \lambda D_{f,K}(J_s, J_t) + \rho M_{f,K}(D_s, D_t) \quad (1)$$

where K is the kernel function induced by ϕ such that $K(\mathbf{x}_i, \mathbf{x}_j) = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$. Basically, the objective function is designed so that the learned adaptive classifier achieves the following three sub-objectives:

- minimizing the loss function on the source domain, which is the first part of Eq. (1), i.e. $\sum_{i=1}^{n} l(f(\mathbf{x}_i), y_i) + \eta||f||_K^2$,
- minimizing the distribution differences between the two domains, which is the second part of Eq. (1), i.e. $D_{f,K}(J_s, J_t)$, and
- minimizing the manifold inconsistency on both source and target data, which is the third part of Eq. (1), i.e. $M_{f,K}(D_s, D_t)$.

In Eq. (1), η, λ and ρ are regularization parameters used to control the contributions of the three sub-objectives which will be discussed as follows.

The first sub-objective is the loss function on the source domain which is minimized by using the structural risk minimization principle [37]. By adopting the Representer theorem [38], the parameters of the classifier can be expressed as $\mathbf{w} = \sum_{i=1}^{n+m} \alpha_i^T \phi(\mathbf{x}_i)$. Following the expansion, the classifier can be rewritten as $f(\mathbf{x}) = \sum_{i=1}^{n+m} \alpha_i^T K(\mathbf{x}_i, \mathbf{x})$. Therefore, learning a classifier f is actually learning a matrix $\boldsymbol{\alpha} \in \mathcal{R}^{(n+m)\times C}$, and the structural risk function can be rewritten as following:

$$loss = \left\|(\mathbf{Y} - \boldsymbol{\alpha}^T\mathbf{K})\mathbf{E}\right\|_F^2 + \eta\text{tr}(\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}) \quad (2)$$

where $\mathbf{K}$ is a kernel matrix corresponding to the function K ($\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$), and $\mathbf{Y}$ is the label matrix. Since the labels on the target domains are not available, the loss is calculated on the source domain only, which is achieved by using the indicator matrix $\mathbf{E}$ with $\mathbf{E}_{ii} = 1$ if $i \leq n$, otherwise $\mathbf{E}_{ii} = 0$. If there are some labeled target instances, their corresponding elements in $\mathbf{E}$ can be set to 1 so the algorithm optimizes $\boldsymbol{\alpha}$ such that the loss on both source and labeled target instances is minimized.

The second sub-objective is to minimize the data distribution difference $D_{f,K}(J_s, J_t)$, which can be measured by

Maximum Mean Discrepancy (MMD) [39]. Formally, the difference can be computed by adding the difference of the marginal distributions and the difference of the conditional distributions:

$$D_{f,K}(J_s, J_t) = D_{f,K}(P_s, P_t) + \sum_{c=1}^{C} D^{(c)}_{f,K}(Q_s, Q_t) \quad (3)$$

Each component in Eq. (3) is calculated by the following equation:

$$D_{f,K}(P_s, P_t) = \left\| \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i) - \frac{1}{m}\sum_{j=n+1}^{n+m} f(\mathbf{x}_j) \right\|^2_{\mathcal{H}}$$

$$D^{(c)}_{f,K}(P_s, P_t) = \left\| \frac{1}{n^{(c)}}\sum_{\mathbf{x}_i \in D_s^{(c)}} f(\mathbf{x}_i) - \frac{1}{m^{(c)}}\sum_{\mathbf{x}_j \in D_t^{(c)}} f(\mathbf{x}_j) \right\|^2_{\mathcal{H}}$$

where $D_s^{(c)}, D_t^{(c)}$ represent the source and target instances belonging to the class c, and $n^{(c)} = |D_s^{(c)}|, m^{(c)} = |D_t^{(c)}|$. Using the expansion $f(\mathbf{x}) = \sum_{i=1}^{n+m} \alpha_i K(\mathbf{x}_i, \mathbf{x})$ and the MMD matrix $\mathbf{M}$, the domain difference can be formularized as:

$$D_{f,K}(J_s, J_t) = \text{tr}(\boldsymbol{\alpha}^{\text{T}}\mathbf{KMK}\boldsymbol{\alpha}) \text{ with } \mathbf{M} = \sum_{c=0}^{C} \mathbf{M}_c \quad (4)$$

where

$$(\mathbf{M}_c)_{ij} = \begin{cases} \frac{1}{n^{(c)}n^{(c)}}, & \mathbf{x}_i, \mathbf{x}_j \in D_s^{(c)} \\ \frac{1}{m^{(c)}m^{(c)}}, & \mathbf{x}_i, \mathbf{x}_j \in D_t^{(c)} \\ \frac{-1}{n^{(c)}m^{(c)}}, & \begin{cases} \mathbf{x}_i \in D_s^{(c)}, \mathbf{x}_j \in D_t^{(c)} \\ \mathbf{x}_j \in D_s^{(c)}, \mathbf{x}_i \in D_t^{(c)} \end{cases} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

Note $n^{(0)} = n, m^{(0)} = m, D_s^{(0)} = D_s, D_t^{(0)} = D_t$.

The last sub-objective exploits the manifold assumption [38] to achieve a better classifier. The assumption is that if two points $\mathbf{x}_i$ and $\mathbf{x}_j$ are geometrically close, they should have similar conditional distribution. In other words, the two instances should belong to the same class. Such assumption can be expressed by the following equation:

$$\begin{aligned} M_{f,K}(D_s, D_t) &= \sum_{i,j=1}^{n+m} f(\mathbf{x}_i) L_{ij} f(\mathbf{x}_j) \\ &= \text{tr}(\boldsymbol{\alpha}^T \mathbf{KLK}\boldsymbol{\alpha}) \end{aligned} \quad (6)$$

where $\mathbf{L}$ is the graph Laplacian matrix which can be calculated by the following formula:

$$L_{ij} = \begin{cases} cosine(\mathbf{z}_i, \mathbf{z}_j) & , \mathbf{z}_i \in \mathcal{N}_p(\mathbf{z}_j) \text{ or } \mathbf{z}_j \in \mathcal{N}_p(\mathbf{z}_i) \\ 0 & , \text{otherwise} \end{cases} \quad (7)$$

where $\mathcal{N}(\mathbf{z}_i)$ denotes the set of p-nearest neighbors to point $\mathbf{z}_i$. In this work, p is set to 10 as recommended in MEDA [9].

By substituting Eq. (2), (4), (6) to Eq. (1), the goal is to find a matrix $\boldsymbol{\alpha}$ to minimize the following function:

$$\begin{aligned} Fitness = &\left\| (\mathbf{Y} - \boldsymbol{\alpha}^T\mathbf{K})\mathbf{E} \right\|_F^2 + \\ &\text{tr}(\eta\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{K}(\lambda\mathbf{M} + \rho\mathbf{L})\mathbf{K}\boldsymbol{\alpha}) \end{aligned} \quad (8)$$

By setting derivative of the objective function to 0, we obtain the solution:

$$\boldsymbol{\alpha} = ((\mathbf{E} + \lambda\mathbf{M} + \rho\mathbf{L})\mathbf{K} + \eta\mathbf{I})^{-1}\mathbf{EY}^T \quad (9)$$

As can be seen in the equation, $\boldsymbol{\alpha}$ depends on $\mathbf{Y}$, but $\mathbf{Y}$ is determined by $\boldsymbol{\alpha}$. To solve the problem, an iterative optimization process is adopted. Firstly, a standard classification algorithm is trained on the source data. The obtained classifier is used to obtain pseudo-labels $\mathbf{Y}$. From the pseudo labels, a matrix $\boldsymbol{\alpha}$ is calculated by Eq. (9). Based on the obtained $\boldsymbol{\alpha}$, new pseudo-labels $\mathbf{Y}$ are generated. Such a procedure is repeated until a maximum number of iterations is reached. It has been shown that ARTL [8] achieves promising results especially on datasets that are difficult to classify by standard classification algorithms (extremely low accuracy). MEDA [9] extends ARTL by applying GFK [32] as a preprocessing step to align the feature space, i.e. reducing the marginal distribution difference.

However, as a gradient-based approach, MEDA converges prematurely to local optima. To address the above limitation, P-MEDA [10] hybridizes population-based and gradient-based mechanisms to evolve a transfer ensemble classifier. The main idea of P-MEDA is to have each population member starts from a promising point in the search space. At each point, gradient descent is applied to achieve a local optimum representing a promising classifier. Once the local optimum is reached, P-MEDA re-initializes the corresponding population member by using the local optima discovered by other population members. After a certain number of generations, P-MEDA then combines all the obtained local classifiers to form an ensemble classifier. Experimental results show that the hybridization can generate better classifiers than MEDA. However, since the task is to optimize a high-dimensional matrix $\boldsymbol{\alpha}$, the main component of P-MEDA is still the gradient-based mechanism (as shown in Eq. (9)) and the communication between population members is limited (mainly in the re-initialization process). Besides, due to the unbounded values in the matrix $\boldsymbol{\alpha}$, P-MEDA relies on a predefined set of classifiers to initialize the population. In order to ensure the diversity of the evolved ensemble classifier, a diverse set of classifiers is required. However, due to the limited number of classifiers, the population size of P-MEDA has to be small. In this work, we address the above limitations by introducing a discrete representation with much lower dimensionality.

III. Proposed Algorithm

Designing a population-based algorithm requires answers for three essential questions: how to describe candidate solutions (i.e. representation), how to evaluate the goodness of candidate solutions (i.e. evaluation), and how to generate promising solutions based on the current solutions (i.e. evolutionary mechanism). This section describes the proposed transfer learning algorithm by answering the three questions.

A. Representation

In most existing transfer classifier induction approaches, the classifier is represented by a matrix $\boldsymbol{\alpha}$, which is an $(n+m) \times C$

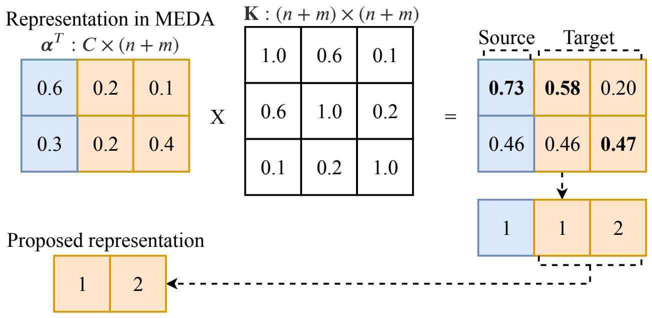


Fig. 1: Representation of transfer classifier induction for a domain adaptation problem where the number of class C = 2, the number of source instances n = 1, and the number of target instances m = 2. MEDA [9] optimizes the $\boldsymbol{\alpha}$ matrix. The target labels are obtained by multiplying $\boldsymbol{\alpha}$ (size $(n+m)\times C$) with the kernel matrix $\mathbf{K}$. In contrast, the proposed representation directly encodes the target labels as an integer vector (size m), which does not require an additional matrix multiplication and has much lower dimensionality.

matrix. To classify the target instances, a matrix multiplication, $\boldsymbol{\alpha}^T \times \mathbf{K}$, is needed to obtain a $C \times (n+m)$ matrix in which each column shows the likelihood that each instance belongs to each of the C class labels. Hence, the matrix's values can be considered a set of weights which combines the kernel values (similar to distances between any pairs of instances) of the kernel matrix $\mathbf{K}$. Finally, each instance is assigned to the most likely class label. An example of the above process is shown in Fig. 1. Thus, the labels for target instances (called target labels) are the final output. Existing algorithms focus on optimizing $\boldsymbol{\alpha}$. This optimization is challenging since $\boldsymbol{\alpha}$ is a high-dimensional matrix. The second problem is for each target label, there are infinitely many $\boldsymbol{\alpha}$ resulting in the same target labels. Hence, although there are finite cases to label the target instances, optimizing $\boldsymbol{\alpha}$ has to explore an infinite search space. Also, such representation requires a constraint that $\boldsymbol{\alpha}$ should correctly classify the source instances, which is the first component in Eq. (1), i.e. $\sum_{i=1}^{n} l(f(\mathbf{x}_i), y_i) + \eta||f||_K^2$. This constraint makes the optimization more complex.

The solution is, instead of optimizing $\boldsymbol{\alpha}$, the target labels are optimized directly. Particularly, a candidate solution is represented by a vector with m elements, as shown in the lower part of Fig. 1. Each element corresponds to a target instance, and its value shows which class the instance is assigned to. In this case, the search dimensionality is significantly reduced from $(n+m)\times C$ to m. Furthermore, the number of possible candidate solutions is finite since each element has C possible values. Such representation can also reduce the computational complexity since it does not require an additional matrix multiplication to obtain target labels as optimizing $\boldsymbol{\alpha}$. Note that the proposed representation is designed to directly learn the target labels, which does not need to learn $\boldsymbol{\alpha}$ as an intermediate step. Thus, the proposed algorithm searches directly the original discrete search space, while existing unsupervised domain adaptation algorithms search the intermediate continuous search space of $\boldsymbol{\alpha}$ before outputting the target labels. In combination with an evolutionary search (which will be showed later), the algorithm is more likely to achieve the global optimal solution or a close-to-optimal solution. The classifier represented by the proposed algorithm assigns the source labels to the source instances, so the source labels are not included in the representation. Hence, using the proposed representation removes the constraint of correctly classifying the source instances, thus the source classification performance in Eq. (1) can be safely removed. Thereby, the fitness function becomes simpler which will be explained in the following subsection. Although the proposed representation is simple, it has not been used in existing transfer classifier induction approaches since its elements are integer values. Dealing with such discrete optimization is not an easy task, especially when mathematical programs, more specifically gradient-based techniques, are used. However, genetic operators of population approaches can naturally cope with such representations, which will be explained later.

B. Fitness Function

The new representation directly assigns class labels to the target instances, whereas the class labels of the source instances are already known. These source labels are not included in the proposed representation. The classifier represented by the proposed representation always has 100% accuracy on the source domain, which allows us to remove the component related to the source classification performance in Eq. (1), i.e. $\sum_{i=1}^{n} l(f(\mathbf{x}_i), y_i) + \eta||f||_K^2$. With the new representation, the optimization goal is to minimize the two remaining components: $D_{f,K}(J_s, J_t)$ — the distribution difference, and $M_{f,K}(D_s, D_t)$ — the manifold inconsistency.

A straightforward way to optimize the two above components is to linearly combine them as in Eq. (1). However, such a combination requires a proper setting for the regularization parameters (λ and ρ) since the two components have different ranges on different datasets. It is not trivial to find the proper setting since different datasets have different scales and there are no labeled instances in the target domain to tune the parameters. We propose to optimize the two components by adopting a hybrid optimization process combining an evolutionary search and a gradient-based search. One component is used as the main fitness function of the evolutionary search to evaluate the candidate solutions. The top solutions generated by the evolutionary search are refined by a gradient-based local search guided by the remaining component.

We select the manifold consistency as the main fitness function since optimizing the manifold consistency can partially reduce the distribution difference. The manifold assumption states that if two data points, x_s and x_t are geometrically close given similar marginal distributions $P_s(x_s)$ and $P_t(x_t)$, their conditional distributions, $Q_s(y_s|x_s)$ and $Q_t(y_t|X_t)$, should be similar [40]. Therefore, the new fitness function should minimize the manifold inconsistency $M_{(f,K)}(D_s, D_t)$ which was defined as $\text{tr}(\boldsymbol{\alpha}^T\mathbf{KLK}\boldsymbol{\alpha})$ in Eq. (6).

However, under the proposed representation, there is no explicit $\boldsymbol{\alpha}$. Each candidate solution $\boldsymbol{z}$ directly represents the target labels, which is the result of $(\boldsymbol{\alpha}^T\mathbf{K})$. Thus, we can rewrite the new fitness function of a candidate solution $\boldsymbol{z}$ as:

$$Fitness(\boldsymbol{z}) = \text{tr}(\mathbf{Y_z}^T\mathbf{L}\mathbf{Y_z}) \tag{10}$$

where $\boldsymbol{Y_z}$ is an $(n+m)\times C$ label matrix defined by $\boldsymbol{z}$, which can be seen as following:

$$(\boldsymbol{Y_z})_{ij} = \begin{cases} 1 & ,(i<n \ \wedge \ (\boldsymbol{y_s})_i = j) \vee (i \geq n \ \wedge \ (\boldsymbol{z})_i = j) \\ 0 & ,\text{otherwise} \end{cases}$$

Basically, $\boldsymbol{Y_z}$ is an one-hot encoded matrix, where each row corresponds to an instance. The row's element corresponding to the instance's label is set to 1, and the other elements are set to 0. If the instance is from the source domain ($i<n$), its label is defined by the source label vector ($\boldsymbol{y_s}$). If the instance is from the target domain ($i \geq n$), its label is defined by the candidate solution $\boldsymbol{z}$.

In comparison with the original objective (Eq. (1)), the proposed fitness function does not need to minimize the loss function on the source domain. The main focus is to minimize the manifold inconsistency, while the distribution difference is minimized by a local search operator that is explained in the following subsection. In comparison with the fitness function of existing unsupervised domain adaptation approaches (Eq. (8)), the proposed fitness function has the following advantages over the original fitness function:

- it does not need to explicitly calculate the matrix $\boldsymbol{\alpha}$ (Eq. (9)), which avoids the matrix inversion operator,
- it obtains the class label directly from the new representation, which reduces the number of matrix multiplications,
- it removes the regularization parameter ρ in Eq. (1),
- it does not need to consider the loss in the source domain since the new representation always classifies the source instances using their existing labels.

In general, the overall goal of the proposed fitness function is to reduce the number of matrix operations, so that the fitness function is less computationally intensive, but still computes the manifold inconsistency and evolves accurate target labels.

C. Population Evolution

In this subsection, we will discuss how to generate a new and promising population based on the previous population. We utilize two well-known genetic operators from GAs: crossover and mutation. We also propose a local search to minimize the distribution difference, i.e. $D_{f,K}(J_s, J_t)$ in Eq. (1). The details of each operator can be seen below.

1) Crossover: A genetic operator that combines the genetic information from two parents to generate new offspring. To be selected as parents, all the population members have to undergo a selection process where the fitter member has more chance to be selected. After selecting two parents, the elements from the two parents are exchanged to form new offspring. In this work, we utilize a single-point crossover, where all elements beyond a randomly picked point is swapped. It is expected that performing the crossover operator between two good parents leads to promising offspring. An example of applying the crossover operator is given in Fig. 2.

2) Mutation: Another genetic operator that aims to preserve and/or enhance the population diversity. When the parents become similar, crossover becomes less useful since the generated offspring are also similar. The mutation operator prevents such situations by altering some elements of each population member, so the population members are less similar, i.e. improving the diversity. Similar to crossover, the fitter population member has more chance to be selected as the parent of mutation. Note that mutation requires only one parent.

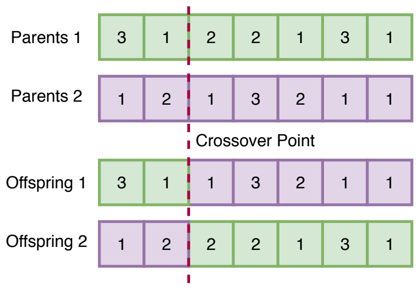


Fig. 2: An example of crossover, where the number of target instances m is 7, the number of class labels C is 3.

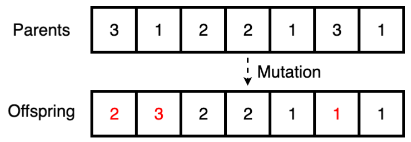


Fig. 3: An example of mutation, where the altering probability is set to C/m (C=3, m=7). The first, second, and sixth elements are mutated (marked in red).

In this work, we adopt a common mutation, called bit-wise mutation in which each element has a probability of being altered. The common setting for the altering probability is $1/m$, if each element is a binary value and m is the total number of elements. In this work, since each element has C possible values (class labels), the altering probability is set to C/m with an expectation of preserving the population diversity. In the proposed representation, each element is a class label of a target instance. If an element is mutated, its value will be changed to another class label. It can be seen that the mutation operator is suitable for the proposed representation since each element has finite possible values (the number of class labels). An example of applying the mutation operator is given in Fig. 3.

3) Local search for reducing distribution difference: Since the fitness function focuses on the manifold consistency, we propose a gradient-based local search to explicitly reduce the domain difference. The main idea is to select the top population members and modify them so that they also have small domain differences. In each iteration, the local search is performed after the two evolutionary operators (crossover and mutation) are finished. Particularly, for each selected member $\boldsymbol{z}$, its corresponding label matrix $\boldsymbol{Y_z}$ is built as in Eq. (10). Based on $\mathbf{Y_z}$, a gradient step is applied to generate the corresponding $\boldsymbol{\alpha_z}$, which is similar to Eq. (9), except that the manifold component is not included. Thus, $\boldsymbol{\alpha_z}$ can be obtained by the following equation:

$$\boldsymbol{\alpha_z} = ((\mathbf{E} + \lambda\mathbf{M})\mathbf{K} + \eta\mathbf{I})^{-1}\mathbf{E}\boldsymbol{Y_z}^T \qquad (11)$$

Note that, although the main goal of local search is to reduce the distribution difference, the generated $\boldsymbol{\alpha_z}$ needs to ensure that the loss on the source domain is also minimized. The main reason is that $\boldsymbol{\alpha_z}$ does not guarantee the source labels are correctly predicted as in the proposed representation. Therefore, the loss on the source domain is still used to generate $\boldsymbol{\alpha_z}$. From the obtained $\boldsymbol{\alpha_z}$, new pseudo labels of the target instances can be formed using the matrix $\boldsymbol{\alpha_z}^T\mathbf{K}$. Finally, the new target labels are used to replace the selected member $\boldsymbol{z}$. As explained before, the manifold consistency and the conditional distribution difference are partially related. Performing conditional distribution alignment is likely to improve the solution's manifold consistency. In the worst case where the local search causes a candidate solution to have worse manifold consistency, such solution will be eliminated by the selection process in the evolutionary process. Hence, it is expected that the proposed hybrid optimization process can lead to the optimal solution which has low manifold inconsistency and low distribution discrepancy.

D. Overall Algorithm

The proposed algorithm starts with an *initialization* step, where each individual is randomly initialized. As discussed in Subsection III.A, each individual can be considered labels of target instances. To initialize an individual, a random class label is randomly picked for each instance. After finishing the *initialization* step, the population is evaluated using Eq. (10).

Then, a new population is generated based on the current population. Firstly, a *selection* mechanism is applied to select parents for the *crossover* and *mutation* operators. The *selection* is designed so that the fitter individuals are more likely to be selected. The idea is to expect promising offspring generated from good parents. The algorithm also adopts *elitism*, which means the best solution from the current generation is copied directly to the new population. The *elitism* is to ensure that the new population is at least as good as the current population. The above steps are basic and essential in genetic algorithms.

The next step is to apply the proposed local search operator to reduce the distribution difference. Firstly, the newly generated population is evaluated by Eq. (10). Based on the fitness values, the top best individuals of the population are selected to undergo the local search at each iteration. The number of top individuals balances the evolutionary search and the gradient-based search. In this work, the gradient-based method is applied to the top 10% of the population. The first reason is that we would like to reduce the distribution differences of the individuals which already have low manifold inconsistencies. It is not worth performing the gradient-based method on solutions with high manifold inconsistency since the final goal is to have a solution which has low distribution difference and manifold inconsistency. Secondly, applying the gradient-based method to a large number of individuals may significantly reduce the population diversity which is likely to result in a local optimal solution. The solutions obtained by the gradient-based search replace their parents in the population.

The process "$selection \rightarrow crossover \rightarrow mutation \rightarrow$ *local search*" is repeated until a maximum number of iterations is reached. The final best solution is selected as the class labels of the target instances. Since the proposed algorithm is designed based on genetic algorithms and MEDA [9], we called the proposed algorithm G-MEDA. The overall process of G-MEDA can be seen in Fig. 4.

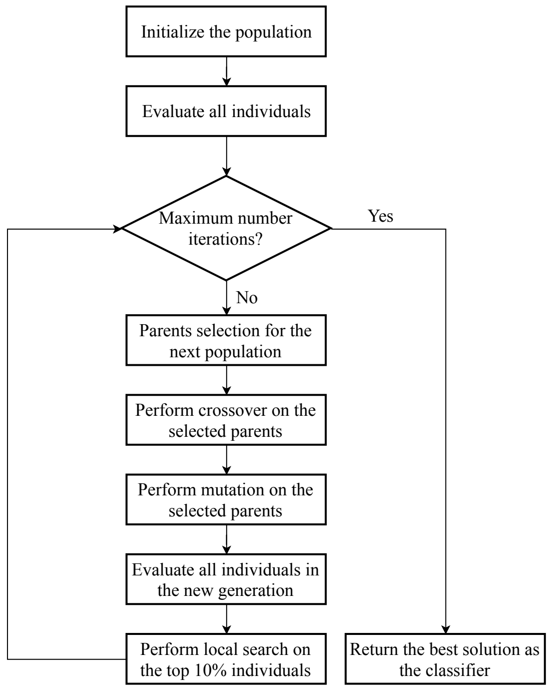

Initialize the population
Evaluate all individuals
Maximum number iterations?
Yes
No
Parents selection for the next population
Perform crossover on the selected parents
Perform mutation on the selected parents
Evaluate all individuals in the new generation
Perform local search on the top 10% individuals
Return the best solution as the classifier

Fig. 4: Overall process of G-MEDA.

IV. Experimental Designs

A. Benchmark Datasets

G-MEDA is tested on three real-world datasets: Amazon Review [41], Office+Caltech10 [42], [31], and Office-31 [43]. These datasets have been widely used as benchmark problems for many domain adaptation algorithms [9], [30], [32]. Table II lists the details of each of these datasets.

Amazon Reviews contains almost reviews for four different types of products from Amazon.com: Books (B), DVDs (D), Electronic (E), and Kitchen appliances (K). Each review can be negative (3 stars or lower) or positive (higher than 3 stars) [41], [44] Similar to Office+Caltech10, we can form 12 domain adaptation cases for this dataset.

Office+Caltech10 contains four different image domains: Amazon (A), DSLR (D), Webcam (W), and Caltech-256 (C). The four domains have the same task which is to classify an image to an office item (in total there are 10 item types). For Office+Caltech10, 800 SURF features are extracted from each image. By selecting two different domains as the source domain and target domain, respectively, we can form 3 $\times$ 4 = 12 domain adaptation cases, such as A$\rightarrow$C, A$\rightarrow$D, ..., W$\rightarrow$D.

Office-31 has three image domains: Amazon (A), DSLT (D), and Webcam (W) with 31 categories. We follow the convention of [45] to perform the experiments, where Resnet-50 is trained on the source domain, and 2048 features are extracted from the obtained model. Based on this dataset, we can form 3 $\times$ 2 = 6 domain adaptation cases.

TABLE II: Domain adaptation datasets.

Dataset	**#Sample**	**#Feature**	**#Class**	**Domains**
Amazon Review	7,996	400	2	B, D, E, K
Office-10	1,410	800	10	A, W, D
Caltech-10	1,123	800	10	C
Office-31	4,110	2048	31	A , W , D

TABLE III: Comparisons with three standard classifiers.

Datasets	Cases	1NN	RF	SVM	G-MEDA
Amazon Review	B → D	55.33	73.09	73.14	**75.29**
	B → E	58.56	74.07	72.67	**76.35**
	B → K	56.68	76.74	77.14	**78.44**
	D → B	56.50	**74.25**	71.10	71.05
	D → E	58.21	75.18	74.32	**76.74**
	D → K	56.68	76.74	77.14	**78.45**
	E → B	55.40	**68.05**	67.70	67.35
	E → D	56.13	69.08	69.63	**72.14**
	E → K	65.13	81.74	81.69	**83.14**
	K → B	56.45	66.85	**70.60**	67.81
	K → D	59.53	67.63	70.69	**72.34**
	K → E	64.51	76.68	78.18	**78.62**
Office+Caltech10	A → C	26.00	33.93	7.57	**47.17**
	A → D	25.48	33.76	6.37	**45.73**
	A → W	29.83	33.22	9.15	**49.41**
	C → A	23.70	40.40	9.60	**57.64**
	C → D	25.48	37.58	7.64	**54.76**
	C → W	25.76	35.93	9.83	**53.13**
	D → W	63.39	44.75	10.17	**90.88**
	D → A	28.50	25.68	10.44	**43.52**
	D → C	26.27	26.09	11.40	**33.17**
	W → A	22.96	28.91	10.33	**42.57**
	W → C	19.86	22.35	11.84	**31.01**
	W → D	59.24	63.06	14.01	**91.23**
Office-31	A → D	79.12	62.85	3.61	**87.65**
	A → C	75.85	62.26	3.14	**87.47**
	D → A	60.17	41.29	3.55	**71.40**
	D → W	95.97	66.16	3.77	**97.38**
	W → A	59.92	46.57	3.51	**72.56**
	W → D	99.40	71.29	4.42	**99.70**

B. Benchmark Techniques

Firstly, we compare the performance of G-MEDA with three standard classification algorithms: KNN where K=1 (1NN), random forest (RF) where the number of trees is set to 128 [46], and support vector machines (SVM) with the RBF kernel. The three classification algorithms are trained directly on the source data and then applied to the target data. We also compare G-MEDA with seven state-of-the-art traditional domain adaptation methods and four deep domain adaptation methods. The comparisons are based on the accuracy of the target domain.

The seven state-of-the-art traditional (shallow) domain adaptation methods include:

- Transfer component analysis (TCA) [25] which reduces the marginal distribution difference via feature subspace learning.
- Joint distribution alignment (JDA) [26] which reduces both marginal and conditional distribution differences.
- Transfer joint matching (TJM) [47] which reduces the marginal distribution difference by building a new feature subspace and selecting source samples.
- Joint geometrical and statistical alignment (JSGA) [30] which builds two feature subspaces for the source and target domains, where the two subspaces should make the two domains closer.
- Geodesic flow kernel (GFK) [32] which performs manifold feature learning.
- Manifold embedded distribution alignment (MEDA) [9] which builds an adaptive classifier.
- Population-based MEDA (P-MEDA) [10] which adopts the population-based mechanism to build an ensemble adaptive classifier.

The four deep domain adaptation methods include:

- Deep adaptation network (DAN) [16] which reduces the domain difference by embedding multi-kernel MMD into the network.
- Domain adversarial neural network (DANN) [11] which learns domain-invariant “deep” features by adding one domain classifier at the last block.
- Collaborative and adversarial network (CAN) [21] extends DANN [11] by adding one domain classifier to each of the network blocks.
- Joint domain alignment and discriminative feature learning (JDDA) [17] which aims to learn features that are not only domain-invariant but also discriminative.

In terms of parameter settings, for each dataset, we pick only one domain adaptation case to tune the parameters. The obtained parameters are applied to all other cases in the dataset. The regularization parameters are searched in {0.1, 1.0, 10.0}, and the dimensionality of the new feature subspace is searched in {10, 20, 30, ..., 90}, which follows the protocol in [9]. All algorithms use the RBF kernel with a kernel width of 0.5. Results of the deep domain adaptation methods are obtained from their original papers.

In P-MEDA [10], the population size is 10, and the maximum number of iterations is 10. In G-MEDA, the population size is 50, and the maximum number of iterations is 10. Since P-MEDA needs a diverse set of classification algorithms to initialize its population, we keep its population size as investigated in the original paper. A further comparison between G-MEDA and P-MEDA with the same population size can be seen in the supplementary material[1]. Both P-MEDA and G-MEDA are run 30 independent times on each domain adaptation case.

V. Experimental Results

A. Comparison with Standard Classification Algorithms

The comparisons between G-MEDA and the three standard classification algorithms on 30 domain adaptation cases are shown in Table III, where the best accuracy is marked in bold. In comparison with 1NN, G-MEDA achieves significantly better performance on all datasets. On Amazon Review, G-MEDA improves at least 10% accuracy over 1NN on all cases. On Office+Caltech10, G-MEDA’s accuracy is about 22% higher than that of 1NN. The most significant improvement is on the W→D case, where G-MEDA improves more than 30% over 1NN. On the Office-31 dataset, since the features are extracted

[1]https://ecs.wgtn.ac.nz/foswiki/pub/Main/BachNguyen/GMEDASup.pdf

from the Resnet-50 network, 1NN achieves quite a high accuracy (78.41% on average). However, by building adaptive classifiers, G-MEDA can still improve the performance over 1NN. The most significant improvement is 13% on the W→A case. RF and SVM achieves better accuracies than 1NN on Amazon Review, but they are still worse than G-MEDA on most cases. On Office+Caltech10 and Office-31, G-MEDA is better than RF and SVM on all cases. It seems that the domain discrepancy of the two latter datasets is much larger than the Amazon Review dataset, which deteriorates the classification performance of RF and SVM.

In general, the experimental results show that G-MEDA can evolve an adaptive classifier that achieves better performance than standard classification algorithms without any adaptation mechanisms.

B. *Comparison with Traditional Domain Adaptation Methods*

The comparisons on the three datasets: Amazon Review, Office+Caltech10, and Office-31 are shown in Tables IV, V, and VI, respectively. In each domain adaptation case, the best accuracy is marked in bold. As can be seen from three tables, feature-based domain adaptation methods perform well on some specific datasets. For example, TCA and TJM achieve good performance on Office+Caltech10, while JSGA and GFK achieve good performance on Amazon Review. In comparison with such feature-based methods, MEDA seems to achieve better performance on all datasets. Particularly, MEDA is about 10% , 6%, and 4% more accurate than the best feature-based methods on Amazon Review, Office+Caltech10, and Office-31, respectively. The most significant improvement of MEDA over other domain adaptation methods is 14% on the B→K case. The results suggest that it is better to evolve an adaptive classifier instead of using a standard classifier on a set of domain-invariant features.

MEDA and G-MEDA have comparative results on the Office+Caltech10 dataset, where G-MEDA is better than MEDA in six cases. On the Office-31, G-MEDA achieves better accuracy than MEDA on four out of the six cases. Meanwhile, on the Amazon Review dataset, G-MEDA is significantly better than MEDA on 11 out of the 12 cases. It can be seen that G-MEDA achieves more reliable performance than MEDA. The possible reason is in the fitness function, where MEDA needs a parameter to balance between the distribution difference and the manifold consistency. Such parameter can be different for different datasets, or even different cases. In contrast, G-MEDA avoids the parameter by optimizing the two components in two steps, which makes G-MEDA more generalized than MEDA.

We also adopt the Friedman significance test at a significance level of 0.05, to compare between G-MEDA and the other algorithms. The results can be seen in Table VII. “W/D/L” means the number of cases that G-MEDA is statistically significantly better, similar, or worse than a benchmark algorithm. “Ave Rank” shows the average ranking of algorithms on all 30 cases. As can be seen from the table, all feature-based methods are significantly worse than G-MEDA on almost all the 30 cases. G-MEDA builds an adaptive classifier based on the features generated by GFK, while GFK uses 1NN. The results show that G-MEDA significantly outperforms GFK on all 30 cases, which show the importance of building an adaptive classifier. Among the seven benchmark algorithms, P-MEDA achieves the highest rank, which shows that the population-based mechanism is effective in evolving the adaptive classifier. However, in comparison with G-MEDA, P-MEDA is significantly worse on 17 out of the 30 cases.

The results show that the proposed search mechanism and fitness function assist G-MEDA in controlling the distribution difference and manifold consistency more adaptive, which leads to the superiority of G-MEDA in most cases.

C. *Comparison with Deep Adaptation Methods*

The comparison is performed on the Office-31 dataset which has been widely used by many deep adaptation methods. The comparison can be seen in Table VI. Among the four deep methods, DANN [11] and CAN [21] achieve higher accuracies than DAN [16] and JDDA [17]. The major difference between the four algorithms is in the way they handle the domain difference. DAN and JDDA are two discrepancy-based methods which use two traditional divergence measures, MMD and CORAL, respectively. In contrast, DANN and CAN adopt adversarial learning which aims to confuse a domain classifier, i.e. the network cannot distinguish between source instances and target instances. The results show that the adversarial method is more suitable for deep domain adaptation. A possible reason is that the adversarial learning is still based on classification, which is more consistent with the design of deep neural networks for classification than the discrepancy-based methods.

As can be seen from Table VI, G-MEDA achieves better accuracy than DANN and CAN on four out of the six cases. The average accuracy of G-MEDA is about 4% higher than that of DANN and CAN, which illustrates the stability of G-MEDA on different domain adaptation cases. The results suggest that the high performance of deep domain adaptation methods is mainly contributed by the power of the representation learning in Resnet rather than the adversarial learning. The advantage of G-MEDA is that it requires only one fine tune process to obtain its feature inputs (Resnet features). In contrast, the deep methods need to run multiple times to get the optimal parameter setting, which is less applicable than G-MEDA since many real-world applications have very little or even no labeled target instances with which to optimize the parameters.

D. *Computation Time*

The computation time of G-MEDA can be seen in Table VIII. We compare the efficiency of G-MEDA with P-MEDA and MEDA. The first reason is that they are closely related. The second reason is that MEDA is already shown to be as efficient as other benchmark traditional methods [9]. In cases with a small number of instances, such as $D \rightarrow W$, $W \rightarrow D$, it is not much different between G-MEDA and MEDA. When the number of instances is larger, the difference between G-MEDA and MEDA is more visible. It seems that G-MEDA

TABLE IV: Comparisons with traditional domain adaptation methods on Amazon Review.

Cases	TCA	JDA	TJM	JGSA	GFK	MEDA	P-MEDA	G-MEDA
B $\to$ D	58.18	58.38	56.98	58.78	57.23	69.28	70.15	**75.29**
B $\to$ E	60.11	57.01	57.26	59.56	57.16	72.02	73.31	**76.35**
B $\to$ K	59.23	56.38	58.83	58.63	57.88	73.19	72.82	**78.44**
D $\to$ B	53.15	56.50	57.50	55.65	56.20	63.10	65.35	**71.05**
D $\to$ E	57.86	56.41	59.71	59.36	59.26	72.07	73.83	**76.74**
D $\to$ K	59.23	56.38	56.83	58.78	57.88	73.19	75.35	**78.45**
E $\to$ B	53.75	56.75	58.90	58.30	56.65	67.55	**68.57**	67.35
E $\to$ D	58.08	56.13	56.93	58.93	56.03	66.93	68.96	**72.14**
E $\to$ K	64.08	59.68	62.63	66.43	65.88	79.34	80.37	**83.14**
K $\to$ B	55.10	53.60	57.65	58.30	57.15	66.15	66.56	**67.81**
K $\to$ D	58.28	57.03	57.33	61.43	59.38	65.58	66.64	**72.34**
K $\to$ E	64.36	59.71	63.21	65.37	64.51	76.08	77.04	**78.62**
Average	58.45	57.00	58.65	59.96	58.77	70.37	71.58	**74.81**

TABLE V: Comparisons with traditional domain adaptation methods on Office+Caltech10.

Cases	TCA	JDA	TJM	JGSA	GFK	MEDA	P-MEDA	G-MEDA
A $\to$ C	39.72	39.45	39.45	27.96	39.00	44.52	43.28	**47.17**
A $\to$ D	31.21	31.85	44.59	27.39	33.12	45.86	**46.67**	45.73
A $\to$ W	41.02	34.24	43.39	24.07	36.95	**53.22**	47.60	49.41
C $\to$ A	46.66	44.68	43.32	40.61	41.75	56.58	55.82	**57.64**
C $\to$ D	45.86	35.67	45.86	28.66	39.49	50.32	**58.66**	54.76
C $\to$ W	39.32	33.22	46.10	32.54	40.68	53.90	**56.75**	53.13
D $\to$ W	89.83	85.08	88.14	74.58	83.05	87.46	84.14	**90.88**
D $\to$ A	32.46	31.84	32.05	59.71	34.55	40.19	40.08	**43.52**
D $\to$ C	33.93	31.70	28.85	**37.04**	30.28	34.73	35.85	33.17
W $\to$ A	30.06	29.44	29.23	**56.16**	33.51	42.80	42.07	42.57
W $\to$ C	32.32	32.59	32.50	**38.56**	28.32	33.66	34.21	31.01
W $\to$ D	91.10	85.35	85.99	77.07	85.35	89.17	85.73	**91.23**
Average	46.28	42.92	46.62	43.70	43.84	52.70	52.57	**53.35**

TABLE VI: Comparisons with traditional and deep domain adaptation methods on Office-31.

Cases	Traditional methods							Deep methods				G-MEDA
	TCA	JDA	TJM	JGSA	GFK	MEDA	P-MEDA	DAN	DANN	CAN	JDDA	
A $\to$ D	78.92	77.51	83.73	12.05	78.51	85.94	85.74	78.60	79.70	65.90	79.80	**87.65**
A $\to$ W	76.48	74.72	79.62	19.62	74.34	85.91	86.34	80.50	82.00	81.50	82.60	**87.47**
D $\to$ A	65.96	65.46	66.06	68.48	62.58	72.38	72.51	63.60	68.20	**99.70**	57.40	71.40
D $\to$ W	96.73	95.72	95.97	92.70	95.72	96.98	96.73	97.10	96.90	63.40	95.20	**97.38**
W $\to$ A	65.64	64.50	68.19	64.00	63.51	73.27	72.90	62.80	67.40	**98.20**	66.70	72.56
W $\to$ D	99.60	99.60	98.80	86.14	99.40	99.40	99.00	99.60	99.10	85.50	**99.70**	**99.70**
Average	80.55	79.58	82.06	57.17	79.01	85.65	85.54	80.40	82.20	82.40	80.20	**86.03**

TABLE VII: Results of significance tests comparing between G-MEDA and other traditional benchmark algorithms

	TCA	JDA	TJM	JSGA
W/D/L	27/2/1	29/0/1	29/0/1	26/0/4
Ave Rank	5.1	6.7	5.4	5.6

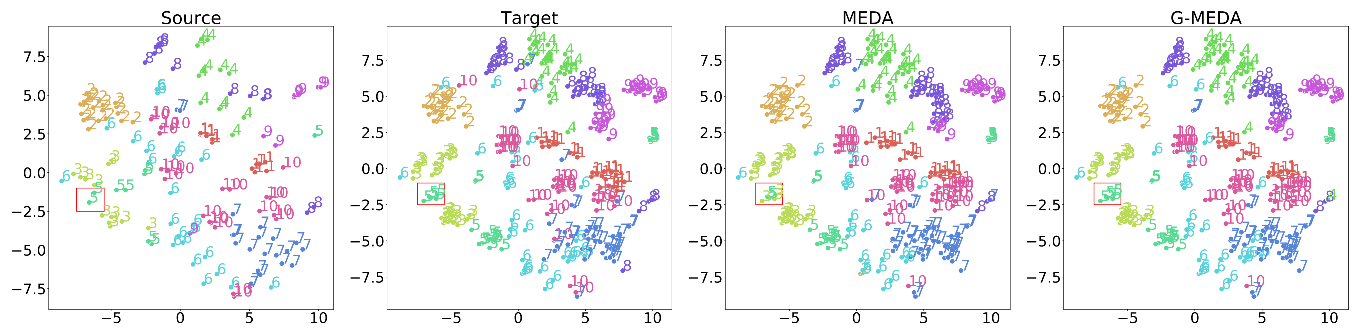
	GFK	MEDA	P-MEDA	G-MEDA
W/D/L	30/0/0	18/6/6	17/6/7	N/A
Ave Rank	6.3	2.6	2.5	**1.8**

is at most four times more expensive than MEDA. However, on a large dataset such as Amazon Review, G-MEDA takes at most 150 seconds (2.5 minutes), which is still quite efficient.

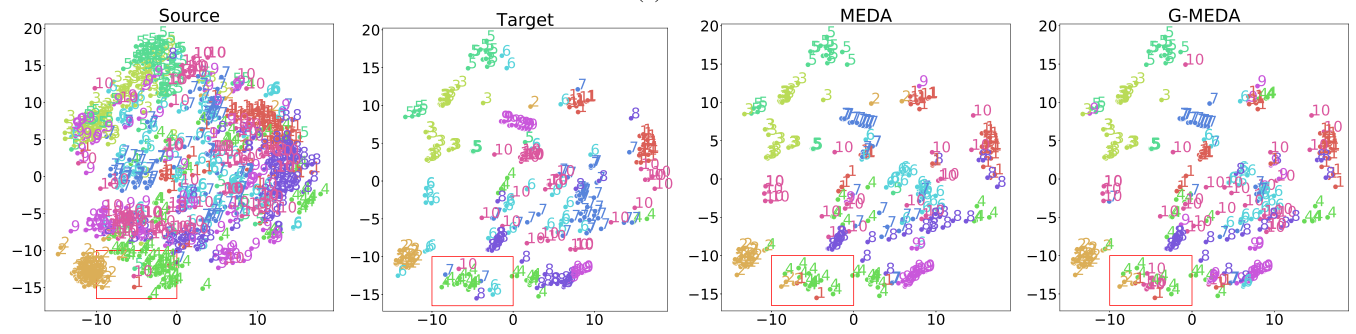
Compared with P-MEDA, G-MEDA is much more efficient. For example, on Office+Caltech10, G-MEDA can be up to ten times more efficient than P-MEDA. Such efficiency of G-MEDA is due to the proposed representation and fitness function. Firstly, representing the classifier by an integer vector is more efficient than using a matrix with continuous values, since it does not require an additional matrix multiplication to obtain labels for the target instances. More importantly, the proposed representation allows G-MEDA to use genetic operators, which is more efficient than P-MEDA that relies heavily on gradient steps consisting of many matrix operators. The main fitness function of G-MEDA (Eq. (10)) also contains a much smaller number of matrix operations than that of P-MEDA (Eq. (8)). Therefore, G-MEDA is computationally much cheaper than P-MEDA even though G-MEDA has a larger number of evaluations than P-MEDA.

E. Analysis on the Adaptive Classifiers

To analyze the obtained classifier, we visualize the source and target data on the Caltech+Office10 dataset. Fig. 5 shows the visualization on two cases, D$\to$W and A$\to$W, where G-MEDA has better and worse performance than MEDA, respectively. The visualization is based on t-SNE [48]. As can be seen from the figure, both G-MEDA and MEDA can successfully transform the two domains such that they have similar marginal and conditional distributions. The accuracies of the two methods on D$\to$W are high (about 90%). The main reason is that both G-MEDA and MEDA use GFK



(a) D $\rightarrow$ W



(b) A $\rightarrow$ W

Fig. 5: Visualization of adaptive classifiers on the Caltech+Office10 dataset.

TABLE VIII: Computation times (in seconds).

Datasets	Cases	MEDA	P-MEDA	G-MEDA
Amazon Review	B $\rightarrow$ D	35.78	454.60	149.38
	B $\rightarrow$ E	36.44	460.16	142.12
	B $\rightarrow$ K	34.97	542.41	135.39
	D $\rightarrow$ B	32.78	542.54	125.39
	D $\rightarrow$ E	33.50	491.86	128.58
	D $\rightarrow$ K	33.52	527.65	128.46
	E $\rightarrow$ B	32.52	479.95	124.05
	E $\rightarrow$ D	33.15	511.27	126.47
	E $\rightarrow$ K	33.52	531.58	128.50
	K $\rightarrow$ B	32.26	520.81	123.25
	K $\rightarrow$ D	31.00	488.24	117.25
	K $\rightarrow$ E	31.64	551.96	120.26
Office+Caltech10	A $\rightarrow$ C	13.52	205.22	55.64
	A $\rightarrow$ D	4.29	94.70	9.54
	A $\rightarrow$ W	5.17	114.14	12.06
	C $\rightarrow$ A	13.22	252.25	44.04
	C $\rightarrow$ D	5.11	139.04	12.58
	C $\rightarrow$ W	6.08	161.69	15.86
	D $\rightarrow$ W	2.46	7.29	3.15
	D $\rightarrow$ A	4.36	25.13	20.25
	D $\rightarrow$ C	5.10	49.44	12.68
	W $\rightarrow$ A	5.00	39.60	12.23
	W $\rightarrow$ C	5.85	59.21	15.37
	W $\rightarrow$ D	2.35	11.10	2.91
Office-31	A $\rightarrow$ D	61.27	5706.80	172.94
	A $\rightarrow$ W	71.20	5957.05	208.23
	D $\rightarrow$ A	66.19	792.03	197.06
	D $\rightarrow$ W	34.49	261.08	33.26
	W $\rightarrow$ A	75.31	1124.53	242.42
	W $\rightarrow$ D	28.24	403.20	38.41

as a pre-processing step to transform the feature space to reduce the distribution difference first. On the D$\rightarrow$W case, GFK can build a feature subspace on which different classes are separable, so the two methods achieve good classification performance. In contrast, G-MEDA and MEDA achieve low classification performance on A$\rightarrow$W. As can be seen in Fig. 5(b), in the source domain, instances from different classes are not clearly separable, which results in poor performance on the target domain. Such results illustrate the importance of the preprocessing step in MEDA and G-MEDA.

MEDA and G-MEDA assign quite similar class labels to the target instances. Their main differences are marked in the red rectangle. On the case D$\rightarrow$W, while G-MEDA classifies all the instances within the red rectangle as class 5, MEDA assigns some instances to class 5 and some other instances to class 3. In the rectangle area of the source domain, there are only two instances from class 5, but there are many instances from the class 3 surrounding the rectangle area. It seems that MEDA focuses much more on the manifold consistency than the conditional distribution difference. In other words, MEDA's decision is affected more by the surrounding instances. Such behavior is because the manifold component usually has much larger values than the distribution difference components. A similar pattern appears in the case A$\rightarrow$W. Particularly, G-MEDA assigns some instances in the red rectangle to the class 10, while MEDA does not assign any instance to the class 10. In fact, in the source domain, the rectangle area contains an instance that is from class 10, so G-MEDA reduces the conditional distribution difference better than MEDA.

The analysis shows that MEDA needs a good parameter setting to control the two objectives, i.e. the manifold consis-

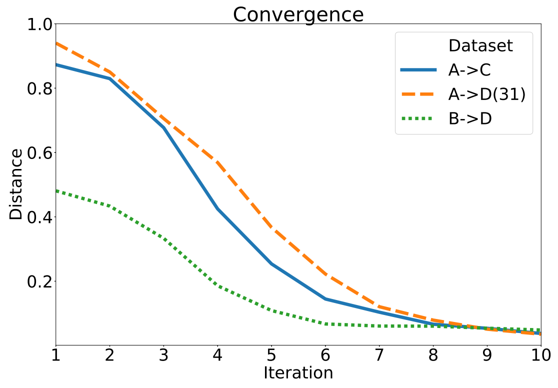


Fig. 6: Convergence curves of G-MEDA.

tency and the distribution difference. However, such a setting is difficult (if not impossible) to obtain since the setting not only depends on the dataset but also changes during the optimization process. G-MEDA achieves both objectives through a hybrid approach of evolutionary-based and gradient-based search, where each search aims to optimize one objective. It can be seen that G-MEDA offers a better approach to handle both objectives, which can be applied to different datasets.

F. Convergence Analysis

We validate the convergence of G-MEDA via the average distance between individuals in the population. The distance between two individuals $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ is calculated by the following formula:

$$d_{ij} = \frac{\mathrm{H}(\boldsymbol{z}_i, \boldsymbol{z}_j)}{m} \tag{12}$$

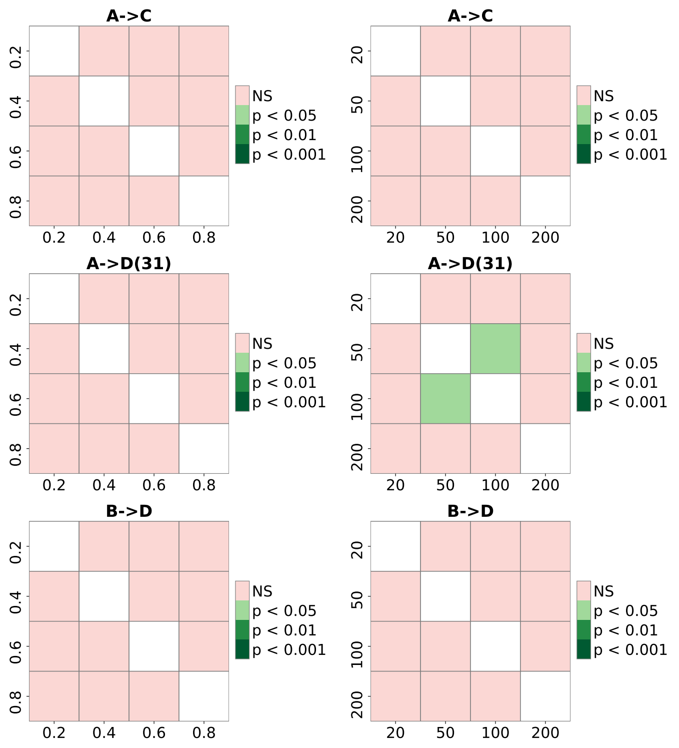
where H(,) is the Hamming distance, m is the length of each individual. Basically, d_{ij} is the ratio between the number of bits where $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ are different and the total number of bits (m), so $d_{ij} \in [0, 1]$. The distance of the whole population P can be obtained by the following formula:

$$Dis(P) = \frac{\sum_{i=1}^{|P|-1} \sum_{j=i+1}^{|P|} d_{ij}}{\frac{|P| \times (|P|-1)}{2}} \tag{13}$$

The distance is designed so that $Dis(P) \in [0, 1]$ for any datasets. The smaller $Dis(P)$, the more converged the population. To examine the convergence of G-MEDA, we record the distance of its population at each iteration. The convergence curves of G-MEDA on three domain adaptation cases $A \rightarrow C$ (Office+Caltecth10), $A \rightarrow D$ (Office31), and $B \rightarrow D$ (Amazon Review) are shown in Fig. 6. It can be seen that G-MEDA can reach a steady state after 10 iterations. The population distance is around 0.03 for the three cases, which means two individuals are different at 3% of all their bits. The results show that G-MEDA can effectively evolve good classifiers in a small number of iterations.

G. Parameter sensitivity analysis

As a population-based algorithm, G-MEDA has three main parameters: the maximum number of iterations, the population



(a) Different mutation rates. (b) Different population sizes.

Fig. 7: Parameter sensitivity: Friedman-Nemenyi post-hoc analysis (best viewed in color). “NS” means “there is no significantly differences”.

size, and the mutation rate (note that the crossover rate is set to (1-mutation rate)). As shown in the previous subsection, G-MEDA converges after 10 iterations which is a reliable setting for the maximum number of iterations. In this section, we further examine the other two evolutionary parameters: mutation rate and population size. The mutation rate varies in {0.2, 0.4, 0.6, 0.8}. The population size varies in {20, 50, 100, 200}. The examination is performed on three representative domain adaptation cases: $A \rightarrow C$ (Office+Caltecth10), $A \rightarrow D(31)$ (Office31), and $B \rightarrow D$ (Amazon Review). The comparisons are performed by the Friedman significance test with a Nemenyi post-hoc analysis and a significance level being set to 0.05.

Fig. 7(a) shows the comparisons between different mutation rates on three representative domain adaptation cases. It can be seen that the four settings of the mutation rate are not significantly different in terms of the obtained accuracies. Thus, the algorithm is not sensitive to the mutation rate. Among the four values, we find that setting the mutation rate to 0.2 results in stably high classification accuracies. Therefore, we recommend setting the mutation rate to 0.2.

Fig. 7(b) shows the comparisons between four different population sizes. The figure shows that there are no significant differences between the four settings of the population sizes. Further analysis of the accuracy (not shown here due to the limited space) reveals the final accuracies are more stable when the population size is set to 200 for domain adaptation cases with large numbers of features like $A \rightarrow D(31)$ (Office31) and

$B \to D$ (Amazon Review). For domain adaptation cases with smaller numbers of features like $A \to C$ (Office+Caltecth10), setting the population size to 100 is sufficient to have stable accuracies. However, if it is important to have a short computation time, one can consider setting the population size to 20 or 50 since the four settings have no significant differences.

VI. Conclusions and Future Work

In this paper, we propose a novel transfer classifier induction approach which hybrids the evolutionary and gradient search to effectively and efficiently evolve a transfer classifier. Firstly, a discrete representation for building an adaptive classifier is proposed to directly learn the target labels. The proposed representation has much smaller dimensionality than the matrix representation which has been widely used by existing transfer classifier adaptation algorithms. The proposed representation is also suitable to genetic operators that have potential global search ability, while existing transfer classifier adaptation algorithms rely on gradient descent with a risk of being trapped at local optima. Along with the novel representation, a novel fitness function is also proposed. The fitness function aims at preserving the manifold consistency which can reduce the conditional distribution difference partially. A gradient-based local search is proposed as an inner step to further reduce the distribution difference. The experimental results on 30 domain adaptation cases show that the proposed algorithms achieve better classification performance than the benchmark traditional and deep domain adaptation methods. Although the proposed algorithm is a population-based domain adaptation method, it is still efficient thanks to its simple fitness function. Besides, it is shown that the proposed hybridization approach can control the manifold consistency and the distribution difference better than combining them in a single fitness function using a regularization parameter.

Although the proposed algorithm achieves good results, there is still room for improvement. The results show that the feature learning step (GFK [32] in this work) has a large impact on classification performance. If the learned features cannot separate source instances from different classes clearly, the proposed algorithm will have poor classification performance on the target domain. Also, the feature learning step is separated from the classifier learning step, which may still limit the interaction between the learned features and the classifier. Potential future work is to integrate the two steps as a single learning step. Such integration would require a new representation that needs to be designed carefully to avoid a large and complex search space. In addition, this work still requires parameters to linearly combine the objectives of unsupervised domain adaptation into a single objective. Such parameters can be removed by considering unsupervised domain adaptation as a multi-objective problem where its objectives are treated separately. In the future, we will investigate employing evolutionary multi-objective optimization methods to achieve unsupervised domain adaptation.

References

[1] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

[2] N. Coudray, P. S. Ocampo, T. Sakellaropoulos, N. Narula, M. Snuderl, D. Fenyö, A. L. Moreira, N. Razavian, and A. Tsirigos, "Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning," *Nature medicine*, vol. 24, no. 10, p. 1559, 2018.

[3] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, p. 115, 2017.

[4] S. J. Pan, Q. Yang *et al.*, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[5] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th International Conference on Machine learning*. ACM, 2007, pp. 193–200.

[6] F. M. Cariucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulò, "Autodial: Automatic domain alignment layers," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5077–5085.

[7] C. Chen, Z. Chen, B. Jiang, and X. Jin, "Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3296–3303.

[8] M. Long, J. Wang, G. Ding, S. J. Pan, and S. Y. Philip, "Adaptation regularization: A general framework for transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1076–1089, 2014.

[9] J. Wang, W. Feng, Y. Chen, H. Yu, M. Huang, and P. S. Yu, "Visual domain adaptation with manifold embedded distribution alignment," in *ACM Multimedia Conference on Multimedia Conference*, 2018, pp. 402–410.

[10] B. H. Nguyen, B. Xue, M. Zhang, and P. Andreae, "Population-based ensemble classifier induction for domain adaptation," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2019, pp. 437–445.

[11] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 1180–1189.

[12] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[13] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 343–351.

[14] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, "Deep reconstruction-classification networks for unsupervised domain adaptation," in *European Conference on Computer Vision*. Springer, 2016, pp. 597–613.

[15] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 136–144.

[16] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, 2015, pp. 97–105.

[17] C. Chao, C. Zhihong, J. Boyuan, and J. Xinyu, "Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation," in *AAAI Conference on Artificial Intelligence*, 2019.

[18] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *European Conference on Computer Vision*. Springer, 2016, pp. 443–450.

[19] X. Peng and K. Saenko, "Synthetic to real adaptation with generative correlation alignment networks," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1982–1991.

[20] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[21] W. Zhang, W. Ouyang, W. Li, and D. Xu, "Collaborative and adversarial network for unsupervised domain adaptation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[22] B. Gong, K. Grauman, and F. Sha, "Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation," in *International Conference on Machine Learning*, 2013, pp. 222–230.

[23] Q. Sun, R. Chattopadhyay, S. Panchanathan, and J. Ye, "A two-stage weighting framework for multi-source domain adaptation," in *Advances in Neural Information Processing Systems*, 2011, pp. 505–513.

[24] Y. Xu, S. J. Pan, H. Xiong, Q. Wu, R. Luo, H. Min, and H. Song, "A unified framework for metric transfer learning." *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 6, pp. 1158–1171, 2017.

[25] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[26] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2200–2207.

[27] S. Si, D. Tao, and B. Geng, "Bregman divergence-based regularization for transfer subspace learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 7, p. 929, 2010.

[28] K. Yan, L. Kou, and D. Zhang, "Learning domain-invariant subspace using domain features and independence maximization," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 288–299, 2017.

[29] S. Wang, L. Zhang, and W. Zuo, "Class-specific reconstruction transfer learning via sparse low-rank constraint," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 949–957.

[30] J. Zhang, W. Li, and P. Ogunbona, "Joint geometrical and statistical alignment for visual domain adaptation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[31] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 999–1006.

[32] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2012, pp. 2066–2073.

[33] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2960–2967.

[34] B. Sun and K. Saenko, "Subspace distribution alignment for unsupervised domain adaptation." in *BMVC*, 2015, pp. 24–1.

[35] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in *Advances in Neural Information Processing Systems*, 2007, pp. 137–144.

[36] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation: Learning bounds and algorithms," *CoRR*, vol. abs/0902.3430, 2009.

[37] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[38] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *International Conference on Computational Learning Theory*. Springer, 2001, pp. 416–426.

[39] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in *Advances in Neural Information Processing Systems*, 2007, pp. 513–520.

[40] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, no. Nov, pp. 2399–2434, 2006.

[41] M. Chen, Z. Xu, K. Q. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," in *Proceedings of the 29th International Conference on Machine Learning*, ser. ICML'12. USA: Omnipress, 2012, pp. 1627–1634.

[42] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[43] B. Sun and K. Saenko, "Subspace distribution alignment for unsupervised domain adaptation." in *BMVC*, vol. 4, 2015, pp. 24–1.

[44] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. 513 – 523, 1988.

[45] J. Zhuo, S. Wang, W. Zhang, and Q. Huang, "Deep unsupervised convolutional domain adaptation," in *Proceedings of the 25th ACM International Conference on Multimedia*, ser. MM '17. New York, NY, USA: ACM, 2017, pp. 261–269.

[46] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?" in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2012, pp. 154–168.

[47] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer joint matching for unsupervised domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 1410–1417.

[48] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

Bach Hoai Nguyen (M'14) received his B.Sc. with First Class Honours and the Ph.D. in computer science at Victoria University of Wellington, New Zealand in 2015 and 2018, respectively. He is currently a Postdoctoral Research Fellow in the School of Engineering and Computer Science at Victoria University of Wellington. His research interests are in evolutionary computation, feature selection, feature construction, and transfer learning.

He is currently the Chair of the IEEE Task Force on Evolutionary Feature Selection and Construction, and a member of the IEEE Computational Intelligence Society.

Bing Xue (M'10) received the B.Sc. degree from the Henan University of Economics and Law, Zhengzhou, China, in 2007, the M.Sc. degree in management from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree in computer science in 2014 at Victoria University of Wellington, New Zealand. She is currently an Associate Professor in School of Engineering and Computer Science at Victoria University of Wellington. She has over 100 papers published in fully refereed international journals and conferences. Her research focuses mainly on evolutionary computation, feature selection, feature construction, image analysis, and transfer learning.

She is currently the Chair of the IEEE Computational Intelligence Society (CIS) Data Mining and Big Data Analytics Technical Committee, Vice-Chair of the IEEE CIS Task Force on Transfer Learning & Transfer Optimization, and Vice-Chair of the IEEE CIS Task Force on Evolutionary Deep Learning and Applications.

Peter Andreae received the B.E. (Honours) degree in electrical engineering from the University of Canterbury, Christchurch, New Zealand, in 1977, and the Ph.D. degree in artificial intelligence from the Massachusetts Institute of Technology, Cambridge, USA in 1985. Since 1985, he has been teaching computer science with the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand.

He is currently an Associate Professor of Computer Science, Associate Dean (Students) and Associate Dean (Academic Development) of the Faculty of Engineering. His research interests include making agents that can learn behavior from experience, but he has also worked on a wide range of topics ranging from reconstructing vasculature from X-rays, clustering algorithms, analysis of microarray data, programming by demonstration, and software reuse.

Mengjie Zhang (M'04-SM'10-F'19) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Hebei, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively. He is currently Professor of Computer Science, Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) in the Faculty of Engineering at Victoria University of Wellington. His current research interests include evolutionary computation with application areas of image analysis, multi-objective optimization, feature selection and reduction, job shop scheduling, and transfer learning. He has published over 500 research papers in refereed international journals and conferences. Prof. Zhang is a Fellow of Royal Society of New Zealand and has been a Panel member of the Marsden Fund (New Zealand Government Funding), a Fellow of IEEE, and a member of ACM.

He was the Chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, and Chair of the IEEE CIS Emergent Technologies Technical Committee and the Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is a Vice-Chair of the IEEE CIS Task Force on Evolutionary Feature Selection and Construction, a Vice-Chair of the Task Force on Evolutionary Computer Vision and Image Processing, and the founding Chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a committee member of the IEEE NZ Central Section.