Alexandria Engineering Journal 133 (2025) 498–516

Contents lists available at ScienceDirect

Alexandria Engineering Journal

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

Original article

OCMTL: Transfer learning by orthogonal core-extraction of a matrix and its application in cross-project defect prediction

Shartaz Sajid Nahid [a,*], Md. Shariful Islam [b], Md. Arman Hossain [c], Muhammad Mahbub Alam [d], Mohammad Shoyaib [a]

[a] Institute of Information Technology, University of Dhaka, Dhaka, Bangladesh
[b] Department of Mathematics, University of Dhaka, Dhaka, Bangladesh
[c] Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh
[d] Department of Computer Science and Engineering, Islamic University of Technology, Gazipur, Dhaka, Bangladesh

ARTICLE INFO

Keywords:
Cross-project defect prediction
Transfer learning
Orthogonal basis core

ABSTRACT

Due to the scarcity of labeled defect data, cross-project defect prediction (CPDP) has become popular nowadays. Most of the successful CPDP methods are based on different transfer learning approaches. However, these methods are mostly sensitive to parameters, dependent on pseudo-labels, and user-defined feature dimensions. To address the issues, we first propose orthogonal core-extraction of a matrix (OCM). OCM is a low-cost and parameter-free iterative method that captures the maximum achievable variance and alleviates the multicollinearity in data by extracting orthogonal cores. We then extend it and propose OCM for transfer learning (OCMTL). OCMTL achieves transfer learning by capturing shared components between source and target without the use of pseudo-labels. Moreover, we propose a method to select the important features. OCMTL is compared with eleven state-of-the-art methods on four benchmark datasets. It achieves 71% average AUC across all datasets, around 2% improvement to its closest competitor (JCSL). Specifically, it achieves average AUC improvements of 4.2% in RELINK against TCA, 1.8% in NASA against JCSL, 0.05% in SOFTLAB against TCA+ and equal average AUC in AEEEM against DMDA_JFR, representing a generalized prediction ability across all datasets. We also conduct significance tests, where OCMTL ranks first in most of the cases.

1. Introduction

Defects in software systems can lead to undesirable results that can cause financial loss, resource wastage, and damage to the reputation of a company. Manual detection of these defects is one of the most resource-intensive and costly tasks of software development [1]. To automatically detect such defects, software defect prediction (SDP) has emerged as an important research topic [2,3]. Most SDP approaches employ machine learning to build defect prediction models using historical data collected within the same project. This approach is referred to as within-project defect prediction (WPDP) [4]. However, new projects are being developed everyday that face a cold start problem without labeled defect data [5]. Therefore, WPDP approaches are unable to build effective prediction models in such scenarios [6].

To address the lack of labeled defect data, cross-project defect prediction (CPDP) has emerged as a potential solution [7]. In CPDP, prediction models are trained using labeled data from a separate software project (source) different from the project of interest (target). Traditional CPDP methods typically use ML-based classifiers [8]. However, these ML-based algorithms generally assume that training and testing data come from the same distribution [9]. In contrast, differences in implementations and complexities among projects in CPDP lead to differences in distributions, which result in degraded prediction performance of these methods [10]. Hence, transfer learning (TL) has emerged as a popular solution in CPDP [10–14] and generally assumes that there exist some common structures between the source and the target domains facilitating a bridge to transfer knowledge. TL-based approaches aim to find some transformations and modify either the source or the target project data or both to reduce the domain gap between them.

CPDP can be divided into two categories: homogeneous CPDP, where source and target projects have the same set of features, and heterogeneous CPDP, where the feature set can be different in type and/or dimension. We are particularly interested in unsupervised homogeneous transfer learning and assume the availability of labeled data only

* Corresponding author.
E-mail addresses: bsse1123@iit.du.ac.bd (S.S. Nahid), mdsharifulislam@du.ac.bd (M.S. Islam), arman.hossain@ewubd.edu (M.A. Hossain), mma@iut-dhaka.edu (M.M. Alam), shoyaib@iit.du.ac.bd (M. Shoyaib).

https://doi.org/10.1016/j.aej.2025.11.039
Received 11 April 2025; Received in revised form 17 October 2025; Accepted 22 November 2025
Available online 26 November 2025

in the source domain but not in the target domain. Existing TL-based CPDP methods can be divided into two categories: data distribution alignment based and subspace alignment based methods [15].

Data distribution alignment based methods aim to reduce the distribution gap in either the marginal distribution (transfer component analysis (TCA) [16], TCA+ [11]) or both marginal and conditional distributions (joint distribution adaptation (JDA) [17], balanced distribution adaptation (BDA) [18]) between the source and target, which is usually measured using maximum mean discrepancy (MMD). In contrast, subspace alignment based methods transform the source and/or target subspaces to bring them closer and measure the alignment (covariance difference minimization, e.g., CORrelation ALignment (CORAL) [19], correlation maximization, e.g., cost-sensitive transfer kernel canonical correlation analysis (CTKCCA) [14]), tradeoff between source–target alignment and source training error, e.g. Joint cross-domain classification and subspace learning for unsupervised adaptation (JCSL) [20]. The many of the aforementioned methods generally use non-linear kernels which improve their performance in many cases but make them resource-intensive and costly. Moreover, both distribution and subspace alignment based methods require parameters, e.g., balance factor μ in BDA and Gaussian kernel parameters in CTKCCA, which make them sensitive to parameter tuning.

Many hybrid methods combine concepts from both distribution and subspace alignment based methods with additional design objectives. Label disentangled analysis (LDA) [15] separates label information from features and performs transfer learning on both feature and label information separately. Visual domain adaptation through locality information (DALI) [21] maximizes target data variance, and minimizes data distribution shifts, subspace divergences and locality projections of source. Low-rank correlation learning (LRCL) [22] separates noise from data to prevent negative transfer and minimizes the distribution gap between the clean source and the target domain data. However, these hybrid TL methods have not been tested in CPDP. Joint feature representation with double marginalized denoising autoencoders (DMDA_JFR) [13] is another hybrid method applied in CPDP which aligns the distributions and improves the discrimination ability using auto-encoders.

Though the use of non-linear kernels in many methods facilitates better alignment between the source and target through data transformation and achieves reasonably good performance, these methods are computationally expensive, and their effectiveness is highly dependent on the exact setting of kernel parameters. Also, the majority of other methods (those that do not use the kernel) require multiple parameters to be tuned to obtain the desired performance, and in many cases, improper parameter settings might lead to a drastically lower performance. Therefore, a parameter-free method is desired, or at least, the performance should not be heavily impacted by the parameter settings.

Moreover, multicollinearity of data is common in CPDP which should be addressed too. It is very important to identify classification-efficient aligned subspaces of the source and the target where the components are independent with high variance, while still retaining the dependency between components that aid classification. Finally, the majority of existing methods only provide an ordered set of features, which not only lacks the mechanism to find the optimal number of features but also fails to provide the proper selection of individual features. Therefore, almost all of the existing methods fail to provide a proper alignment of the subspaces with an appropriate feature size.

To address the aforementioned issues, we aim to design a stable, reliable, and computationally less expensive transfer learning method that can achieve effective subspace alignment of the source and target and extract classification-efficient latent features without requiring extensive parameter tuning. With these desirable properties, we first propose *orthogonal core-extraction of a matrix* (OCM), a second-order representation of the Orthogonal basis-core extraction using Tensor Ring (OTR) [23] which shows its superiority in better classification by decomposing an Nth order tensor in an iterative process. Alongside inheriting all the advantageous properties of OTR in OCM, we demonstrate the convergence of OCM which is inspired by the convergence of tensor decomposition given in [24]. We also propose a TL-based CPDP method using it, namely, *OCM for transfer learning* (OCMTL).

The key contributions of the paper are as follows: (i) orthogonal latent space extraction by OCM with maximum achievable variance, (ii) extension of OCM for transfer learning by capturing the shared subspace between the source and the target with better discrimination ability, and its application in CPDP, (iii) theoretical proof of convergences for both OCM and OCMTL, (iv) alleviation of multicollinearity of the data in both OCM and OCMTL with low-cost and parameter-free mechanisms, (v) selection of appropriate feature dimensions automatically in OCMTL, and (vi) validation of OCMTL and its comparison with state-of-the-art transfer learning methods on four benchmark CPDP datasets with sixty six cross-project pairs.

The rest of the paper is organized as follows. Section 2 reviews the related work on TL and CPDP methods. Section 3 details the preliminaries related to the proposed methods. Section 4 describes the proposed methods in detail. Section 5 provides the experimental design such as dataset description, implementation details, performance metrics and significance test. Section 6 presents and analyzes the results with necessary discussions. Section 7 discusses the threats to validity. Finally, Section 8 concludes the paper with a summary of the work and the future work.

2. Related work

CPDP techniques use the knowledge of labeled source data to predict the labels of the target. Traditional CPDP models commonly use ML-based classifiers, such as Logistic Regression (LR), Naive Bayes (NB), Support Vector Machines (SVM) and Decision Trees, as base models [8]. However, their performance in CPDP is not satisfactory. This is because the source and target data generally differ due to the differences in their implementations and functional complexities. For example, the datasets of two browsing software, namely Firefox and Internet Explorer 8, consist of the same set of features, however, their feature values may be different because of the difference in their development processes. A large-scale experiment is conducted by Zimmermann et al. [6] on 622 cross-project pairs of 12 software projects, where only 3.4% of the prediction tasks achieve acceptable performance, which also shows the challenges posed by the difference between the source and target projects. However, these traditional methods assume that source and target data follow a similar distribution, which is rarely true in practice, making them insufficient to perform well in CPDP.

To minimize the impact of cross-project difference, Turhan et al. [25] proposed a CPDP method that uses a nearest neighbor filter (NN-filter) to select source project instances that are similar to target project instances. Besides this, there are other methods [26,27] that adopt a feature selection strategy to select the best set of features for CPDP. However, selecting features or samples might discard useful information. Therefore, to achieve better performance in CPDP, either the source or target or both need to be transformed so that the difference between the source and target is minimized. To achieve this, transfer learning is commonly applied in CPDP, which can be unsupervised where target labels are unavailable, or semi-supervised where a few target labels are available. In this paper, we focus on unsupervised transfer learning.

Some of the popular data distribution alignment based methods are TCA, TCA+, JDA and BDA. TCA aims to make the marginal distributions of the source and the target project similar by mapping them in a latent space. The authors in [11] observed that the performance of TCA in CPDP is sensitive to normalization techniques, and proposed TCA+ that introduces a heuristic method to choose a suitable normalization based on data distributions. In TPTL [28], it is reported that the performance of TCA+ is dependent on the selection of the

source project, and to overcome this, a two-phase transfer learning model based on TCA+ is proposed. It first selects the source project based on the distribution similarity and then builds the model using TCA+. In practice, the source and target project in CPDP mostly have differences in both marginal and conditional distributions. However, these methods focus on minimizing the difference of only marginal distributions between the source and target.

JDA minimizes both marginal and conditional distributions and is applied in CPDP [12]. It is experimented on four defect datasets and demonstrates better performance than those that only address marginal distributional differences. However, JDA treats marginal and conditional distribution alignment equally, which may result in reduced performance. BDA solves this issue by adjusting the relative importance of marginal and conditional distributions. BDA is also applied in CPDP [10] and evaluated on four defect datasets achieving improved average performance compared to six baseline TL methods. However, BDA highly relies on the balance factor (μ) to adjust the importance. Also, there are other parameters such as the regularization parameter which make the performance dependent on parameter tuning.

The aforementioned methods only preserve the global geometry of data by reducing distribution differences whereas the local relationship of data also needs to be preserved to ensure discriminability. DMDA_JFR [13] uses two auto-encoders to capture both global and local feature representations and uses a repeated pseudo-label strategy to match the distributions progressively. However, the use of auto-encoders with multiple stacked layers can increase the training time. Moreover, to match conditional distributions, JDA, BDA, and DMDA_JFR use pseudo-labels, which can degrade the performance if the predicted pseudo-labels deviate too much from the real labels.

Several TL-based CPDP methods, such as CORAL and CTKCCA, aim to align the source and target subspaces. CORAL is a parameter-free, non-iterative TL method that aims to align the source and target project data by reducing the difference between their covariances. It has also been applied to CPDP in [29,30], which shows the effectiveness of covariance to align the source and target project data. In practice, most of the CDPD datasets contain many more non-defective instances than defective ones. To handle this class imbalance problem, CTKCCA incorporates a cost matrix to penalize misclassifications. To align the subspaces, it transforms both source and target data into a higher-dimensional space using kernel canonical correlation analysis (KCCA) so that the nonlinear correlation between them is maximized. However, its use of predefined cost functions and non-linear transformations may not adapt to data with varying distributions [31]. Additionally, it uses a single kernel to transform the data, which may not capture the correlation accurately [32]. Unlike traditional TL approaches that first seek a domain-invariant subspace and then train a classifier in a separate stage, JCSL [20] (not tested in CPDP) aims to learn both a domain-invariant subspace and a reliable classifier by simultaneously optimizing a trade-off between the alignment and the classification (source training) error.

Some hybrid TL methods (not tested in CPDP) aim to achieve several objectives alongside finding the alignment. LDA decouples label information from features and aligns the label-erased and label-reconstructed features across domains. It also achieves discrimination ability through the cross-association of these features. DALI addresses both marginal and conditional distribution shifts using pseudo-labels, preserves local information via a Laplacian graph-based transformation, and achieves discrimination ability in target by decreasing within-class variance and increasing between-class variance. LRCL aims to alleviate the impact of negative transfer by denoising the data through low-rank learning and learns a common discriminative latent subspace where the distribution difference between clean source and target data is minimized. However, the performance of these methods heavily depends on finding the right settings of the multiple parameters and the accuracy of pseudo-labels.

Most of the aforementioned methods [10,11,14,16,17,21] use kernels and choosing the right kernel with the right parameters is challenging, especially non-linear kernels are computationally costly. Also, all methods except CORAL involve additional parameters that need to be tuned for each new dataset to obtain better results, which is a rigorous task. Moreover, the use of pseudo-labels by many methods [10,13,15,17,21] may make them unreliable. Furthermore, to the best of our knowledge, the selection of the appropriate feature dimension is ignored and the users need to set the right dimension by repetitive experimentation.

In this paper, we focus on designing a simple but effective transfer learning method that aligns the source and the target in a transformed shared low-dimensional subspace. Different from the existing literature, OCMTL is designed to perform without any use of parameters, pseudo-labels, and kernels. This makes our method straightforward to use, consistent, and lightweight in terms of computational resources and runtime. We also propose a feature selection method and integrate it with OCMTL, which further enhances the effectiveness by selecting the appropriate feature dimension automatically without any need for manual intervention.

3. Preliminaries

In [23], Akhter et al. proposed Orthogonal basis-core extraction using Tensor Ring (OTR) for an Nth order tensor, $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, where $I_n, n = 1, 2, \ldots, N$ indicates the respective tensor dimensions, and can demonstrate better discrimination at a lower cost. OTR generates orthogonal basis cores in an iterative manner where the nth basis core is updated by the projection of the mode-n unfolding of $\mathcal{T}$ on the circular product of the remaining $(N-1)$ semi-orthogonal reshaped basis matrices starting from $n+1$, $G(N-1, n)$. The nth basis matrix U_n is updated by assigning the left singular vectors of the projected data to it, and the nth basis core $\mathcal{U}_n$ can be found by reshaping U_n. To understand the whole process, let us consider a 3-rd order tensor $\mathcal{B} \in \mathbb{R}^{m \times n \times p}$ and the mode-1, mode-2 and mode-3 unfoldings of $\mathcal{B}$ are $B_{[1]}^{m \times np}$, $B_{[2]}^{n \times pm}$ and $B_{[3]}^{p \times mn}$, respectively. In OTR, $\mathcal{B}$ is decomposed into three semi-orthogonal basis cores: $\mathcal{U}_1 \in \mathbb{R}^{r_1 \times m \times r_2}$, $\mathcal{U}_2 \in \mathbb{R}^{r_2 \times n \times r_3}$ and $\mathcal{U}_3 \in \mathbb{R}^{r_3 \times p \times r_1}$. The basis cores are obtained by an iterative method, where in each iteration, the nth basis core is updated by assigning the reshaped left singular vectors of the matrix, M_n, given by

$$M_n = B_{[n]}[G(N-1, n)]^T. \tag{1}$$

For example, M_1 can be found, using $B_{[1]}^{m \times np}$ and matricizing $\mathcal{U}_2$ (mode-1 unfolding and reshaping) and $\mathcal{U}_3$ (mode-1 unfolding) into $U_2^{r_2 n \times r_3}$ and $U_3^{r_3 \times pr_1}$, respectively, as follows

$$\begin{aligned} M_1^{m \times r_1 r_2} &= B_{[1]}^{m \times np}[U_2^{r_2 n \times r_3} \times U_3^{r_3 \times pr_1}]^T \\ &= B_{[1]}^{m \times np}[G^{r_1 r_2 \times np}]^T \, (multiplication\ and\ reshaping) \end{aligned} \tag{2}$$

Finally, U_1 is updated by setting the reshaped left singular vectors of M_1 to it. Similarly, U_2 and U_3 are updated. They also show that the basis cores capture the maximum achievable variance of the given data and demonstrate better classification performance. However, the convergence of their iterative process is not shown.

4. Proposed method

In this section, we first propose *orthogonal core-extraction of a matrix* (OCM), an iterative process like OTR, which can be applied to identify defects within projects. OCM, in addition to OTR, provides proof of convergence of the iterative process. To identify cross-project defects, we then extend OCM and propose *OCM for transfer learning* (OCMTL). To select the desired number of features, we also introduce a feature selection method that focuses on maximizing the alignment between the latent feature spaces of the source and target as well as their individual variances in a reduced dimension. The details of the processes are given in the following subsections.

4.1. Orthogonal core-extraction of a matrix (OCM)

It is well known that a given data matrix $X \in \mathbb{R}^{m\times n}$ can be decomposed into a basis matrix $P \in \mathbb{R}^{m\times r}$ and a coefficient matrix $Q \in \mathbb{R}^{n\times r}$ such that $X = PQ^T$ where r is the rank of X. However, in our proposal, we want to identify the core matrices, $U_{P_{|k}} \in \mathbb{R}^{m\times r}$ and $U_{Q_{|k}} \in \mathbb{R}^{n\times r}$, corresponding to matrices P and Q, respectively, where the sub-subscript $|k$ in $U_{P_{|k}}$ and $U_{Q_{|k}}$ (or subscript $|k$ in any matrix) denotes the first k columns of U_P and U_Q (of the corresponding matrix) that capture the most variance of data. Specifically, our objective is to discover a $U_{Q_{|k}}$ so that the original data can be projected onto a reduced-dimensional space to prepare the training and test data with the following properties: (i) preserve the variance of the original data as much as possible to achieve a better discriminability, (ii) alleviate the multicollinearity from the data while preserving the effective interactions between the features, and (iii) remove the uninformative features from the data.

To achieve the aforementioned goals, we propose a second-order version of OTR (originally designed for an Nth order tensor) for the data matrix X, referred to as the *orthogonal core-extraction of a matrix* (OCM). OCM iteratively discovers the desired core matrices of X, denoted by $U_{A_{|k}}$ and $U_{C_{|k}}$ that are $m\times k$ and $n\times k$ matrices, respectively. We can represent both $U_{A_{|k}}$ and $U_{C_{|k}}$ in tensor form with the ring property like OTR as $\mathcal{U}_A^{r_1\times m\times r_2}$ and $\mathcal{U}_C^{r_2\times n\times r_1}$, respectively, where we assume $k = r_1 \times r_2$. Let the mode-1 and mode-2 unfolding of the data matrix X be denoted as $X_{[1]}^{m\times n}$ and $X_{[2]}^{n\times m}$, respectively. Matricizing and reshaping $\mathcal{U}_C$, we can get $U_{C_{|k}}^{r_1r_2\times n}$. Following (2), we can write

$$M_A^{m\times r_1r_2} = X_{[1]}^{m\times n}[U_{C_{|k}}^{r_1r_2\times n}]^T$$

By setting the left singular vectors of M_A to $U_{A_{|k}}$, we can get

$$U_{A_{|k}}^{m\times r_1r_2} = SVD_u(M_A^{m\times r_1r_2}) \tag{3}$$

Similarly, we can get $U_{C_{|k}}$ from $M_C^{n\times r_1r_2} = X_{[2]}^{n\times m}[U_{A_{|k}}^{r_1r_2\times m}]^T$

$$U_{C_{|k}}^{n\times r_1r_2} = SVD_u(M_C^{n\times r_1r_2}) \tag{4}$$

However, for matrix data in OCM, the basis cores in (3) and (4) can be directly represented by (5) and (6), respectively.

$$U_{A_{|k}}^{m\times k} = SVD_u(X^{m\times n}U_{C_{|k}}^{n\times k}) \tag{5}$$

$$U_{C_{|k}}^{n\times k} = SVD_u((X^{m\times n})^TU_{A_{|k}}^{m\times k}) \tag{6}$$

Iterating on (5) and (6) and to obtain $U_{A_{|k}}$ and $U_{C_{|k}}$, the major challenges of OCM are: (i) the convergence of the iterative process defined in (5)–(6) and (ii) the robustness of OCM in the presence of inter-dependent features in data.

We assume that X is standardized and the SVD of it is $A\Sigma C^T$, where the columns of A and C are the left and right singular vectors of X, respectively. We can truncate matrices A and C to get $A_{|k}$ and $C_{|k}$ respectively where $k < n$. Let $A_{|k}^c$ and $C_{|k}^c$ be the matrices consisting of the rest $(n-k)$ columns in those matrices so that the column spaces of $A_{|k}^c$ and $C_{|k}^c$ are the orthogonal complement spaces of the column spaces of $A_{|k}$ and $C_{|k}$, respectively. We facilitate the matrices with the first k components and the matrices with their complement spaces to measure the alignment between our constructed basis cores ($U_{A_{|k}}$ and $U_{C_{|k}}$) and their corresponding true spaces (A and C) which we will discuss later. Here, the alignment of $U_{C_{|k}}$ with $C_{|k}$ is measured by subspace similarity [33] between them and is defined as follows

$$\tan(C_{|k}, U_{C_{|k}}) = \frac{\sin(C_{|k}, U_{C_{|k}})}{\cos(C_{|k}, U_{C_{|k}})} = \frac{\sigma_{\max}(C_{|k}^{cT}U_{C_{|k}})}{\sigma_{\min}(C_{|k}^TU_{C_{|k}})},$$

where i denotes the iteration number. The lesser the angle between the subspaces, the more will be their similarity. Based on the design of OCM, the iterative process can also be expressed as follows

$$U_{A_{|k}}\Sigma_{A_{|k}}V_{A_{|k}}^T = SVD(XU_{C_{|k}}) \tag{7}$$

$$U_{C_{|k}}\Sigma_{C_{|k}}V_{C_{|k}}^T = SVD(X^TU_{A_{|k}}) \tag{8}$$

where $U_{C_{|k}}$ is randomly initialized in a way so that $\tan(C_{|k}, U_{C_{|k}}^0) < 1$. It is expected that the subspace similarity of $U_{C_{|k}}$ with $C_{|k}$ is gradually increased through the iterative process. Convergence of the iterative process is shown in Theorem 4.2. Moreover, through this process, the similarity between $U_{C_{|k}}^{i+1}$ and $U_{C_{|k}}^i$ is also increased gradually (which is discussed later). From (8), we can write

$$\begin{aligned} U_{C_{|k}}^{i+1}\Sigma_{C_{|k}}^{i+1}(V_{C_{|k}}^{i+1})^T &= C\Sigma A^TU_{A_{|k}}^{i+1} \\ C_{|k}^TU_{C_{|k}}^{i+1}\Sigma_{C_{|k}}^{i+1}(V_{C_{|k}}^{i+1})^T &= C_{|k}^TC\Sigma A^TU_{A_{|k}}^{i+1} \\ &= \begin{bmatrix} I_k & \mathbf{0}_{k\times(n-k)} \end{bmatrix}\Sigma A^TU_{A_{|k}}^{i+1} = \Sigma_{|k}A_{|k}^TU_{A_{|k}}^{i+1} \end{aligned} \tag{9}$$

Similarly,

$$(C_{|k}^c)^TU_{C_{|k}}^{i+1}\Sigma_{C_{|k}}^{i+1}(V_{C_{|k}}^{i+1})^T = \begin{bmatrix} \mathbf{0}_{(n-k)\times k} & I_{n-k} \end{bmatrix}\Sigma A^TU_{A_{|k}}^{i+1} = \Sigma_{|k}^cA_{|k}^{cT}U_{A_{|k}}^{i+1} \tag{10}$$

Notice that the column space of $U_{C_{|k}}$ (left singular vectors of $X^TU_{A_{|k}}$) belongs to the column space of X^T according to Lemma 4.1.

Lemma 4.1. *The column space of $U_{C_{|k}}$ is a subspace of the column space of X^T.*

Proof. Since columns of $U_{C_{|k}}$ are the left singular vectors of $X^TU_{A_{|k}}$, they span the same column space of $X^TU_{A_{|k}}$. The ith column vector $(X^TU_{A_{|k}})_i = (C\Sigma A^TU_{A_{|k}})_i = \alpha_{i1}C_1 + \cdots + \alpha_{in}C_n$, where $C_1, C_2, \ldots, C_n$ are the columns of the matrix C and $(\alpha_{i1}, \ldots, \alpha_{in})^T$ is the ith column vector of $\Sigma A^TU_{A_{|k}}$, which indicates that the columns of C span the column space of $X^TU_{A_{|k}}$ and thereby the column space of X^T. Therefore, columns of $U_{C_{|k}}$ belong to the column space of X^T, i.e., the column space of C. □

Theorem 4.2. *The angle between the true subspace $C_{|k}$ and the iteratively constructed approximate subspace $U_{C_{|k}}$ will gradually decrease with iterations.*

Proof. Following the iterative process defined in (7)–(8), we have to show

$$\tan\left(C_{|k}, U_{C_{|k}}^{i+1}\right) \le \left(\frac{\lambda_{k+1}}{\lambda_k}\right)^2 \tan\left(C_{|k}, U_{C_{|k}}^{i}\right), \forall k \ge 0,$$

where λ_k denotes kth singular value of the data. Now the angle between $C_{|k}$ and $U_{C_{|k}}$ after $i+1$ iterations is

$$\begin{aligned} \tan(C_{|k}, U_{C_{|k}}^{i+1}) &= \frac{\sigma_{max}(C_{|k}^TU_{C_{|k}}^{i+1})}{\sigma_{min}(C_{|k}^TU_{C_{|k}}^{i+1})} = \left\|C_{|k}^{cT}U_{C_{|k}}^{i+1}\left(C_{|k}^TU_{C_{|k}}^{i+1}\right)^{-1}\right\|_s \\ &= \left\|C_{|k}^{cT}U_{C_{|k}}^{i+1}\Sigma_{C_{|k}}^{i+1}(V_{C_{|k}}^{i+1})^T\left(C_{|k}^TU_{C_{|k}}^{i+1}\Sigma_{C_{|k}}^{i+1}(V_{C_{|k}}^{i+1})^T\right)^{-1}\right\|_s \\ &\le \frac{\sigma_{max}\left(C_{|k}^{cT}U_{C_{|k}}^{i+1}\Sigma_{C_{|k}}^{i+1}(V_{C_{|k}}^{i+1})^T\right)}{\sigma_{min}\left(C_{|k}^TU_{C_{|k}}^{i+1}\Sigma_{C_{|k}}^{i+1}(V_{C_{|k}}^{i+1})^T\right)} \le \frac{\lambda_{k+1}\sigma_{max}\left(A_{|k}^{cT}U_{A_{|k}}^{i+1}\right)}{\lambda_k\sigma_{min}\left(A_{|k}^TU_{A_{|k}}^{i+1}\right)} \\ &= \frac{\lambda_{k+1}}{\lambda_k}\tan\left(A_{|k}, U_{A_{|k}}^{i+1}\right) \end{aligned}$$

Therefore, $\tan(C_{|k}, U_{C_{|k}}^{i+1}) \le \frac{\lambda_{k+1}}{\lambda_k}\tan\left(A_{|k}, U_{A_{|k}}^{i+1}\right)$. Similarly, we can show that, $\tan\left(A_{|k}, U_{A_{|k}}^{i+1}\right) \le \frac{\lambda_{k+1}}{\lambda_k}\tan\left(C_{|k}, U_{C_{|k}}^{i}\right)$. Therefore,

$$\tan\left(C_{|k}, U_{C_{|k}}^{i+1}\right) \le \left(\frac{\lambda_{k+1}}{\lambda_k}\right)^2 \tan\left(C_{|k}, U_{C_{|k}}^{i}\right),$$

which ensures the convergence. □

It is straightforward according to (9)–(10) and Lemma 4.1 that both $U_{C_{|k}}^{i+1}$ and $U_{C_{|k}}^i$ gradually align with the first k singular vectors of C and thus come closer to each other. This gives us an indication that

we can use the angle between $U^{i+1}_{C_{|k}}$ and $U^{i}_{C_{|k}}$ as a stopping criterion of the proposed iterative process. However, based on Remark 1, we use (11) as the stopping criteria of OCM

$$\|U^{i+1}_{C_{|k}} - U^{i}_{C_{|k}}\|_F \leq \epsilon. \tag{11}$$

Remark 1. At the point of convergence, the specific angle between $U^{i+1}_{C_{|k}}$ and $U^{i}_{C_{|k}}$ for different inputs may vary, and will be difficult to compare with a single threshold. On the other hand, (11) indirectly assures minimum achievable angle between $U^{i+1}_{C_{|k}}$ and $U^{i}_{C_{|k}}$.

After convergence, the k singular vectors of $U_{C_{|k}}$ capture the most variance and remove the uninformative features of the data, and therefore, $U_{C_{|k}}$ can be used to prepare the data in a reduced dimension ($\tilde{X}_{|k} = XU_{C_{|k}}$), where multicollinearity among the columns in $\tilde{X}_{|k}$ is also alleviated as the columns of $U_{C_{|k}}$ are orthogonal. It is noteworthy to mention that OCM can be applied to WPDP at a lower cost. To perform CPDP, we extend OCM and propose *OCM for transfer learning* (OCMTL).

4.2. Orthogonal core-extraction of a matrix for transfer learning (OCMTL)

Suppose we have labeled source data $X_s \in \mathbb{R}^{m_s \times n}$ and unlabeled target data $X_t \in \mathbb{R}^{m_t \times n}$. In CPDP, we learn from X_s and predict the labels of X_t under the assumption that there exists important shared information between them. We extend OCM and propose *OCM for transfer learning* (OCMTL) to capture the common information by discovering a shared subspace between the source and target. For better performance, OCMTL should have the following desired properties:

i. It should automatically identify a shared subspace by aligning the source and target and be able to select the desired number of features.
ii. It should be parameter-free so that a better result can be obtained without the need for a rigorous parameter tuning.
iii. It should achieve a high discrimination ability by capturing the maximum achievable variance, alleviating multicollinearity and removing uninformative features of the given data but retaining the useful dependency information of correlated features along the variance maximizing direction.

To achieve the aforementioned properties, we pre-process X_s and X_t and then decompose them into two sets of basis cores $U_{A_{s_{|k}}} \in \mathbb{R}^{m_s \times k}$ and $U_{C_{s_{|k}}} \in \mathbb{R}^{n \times k}$, and $U_{A_{t_{|k}}} \in \mathbb{R}^{m_t \times k}$ and $U_{C_{t_{|k}}} \in \mathbb{R}^{n \times k}$, respectively, following (7)–(8). The pre-processing involves an optional Log transformation (when the feature data are skewed and are not derived from other features) and a z-score normalization [3,34].

The SVDs of X_s and X_t are $A_s\Sigma_s C_s^T$ and $A_t\Sigma_t C_t^T$ respectively, and their SVDs with the first k columns are $A_{s_{|k}}\Sigma_{s_{|k}}C_{s_{|k}}^T$ and $A_{t_{|k}}\Sigma_{t_{|k}}C_{t_{|k}}^T$ respectively. Let $A^c_{s_{|k}}$, $A^c_{t_{|k}}$, $C^c_{s_{|k}}$ and $C^c_{t_{|k}}$ denote the orthogonal complement spaces of the column spaces of $A_{s_{|k}}$, $A_{t_{|k}}$, $C_{s_{|k}}$ and $C_{t_{|k}}$, respectively, and consist of the remaining $(n-k)$ columns of corresponding true spaces. The matrices with the first k components and their complement spaces facilitate to measure the alignment between each of the constructed basis cores and its related true space (i.e., $U_{A_{s_{|k}}}$, $U_{C_{s|k}}$, $U_{A_{t_{|k}}}$ and $U_{C_{t_{|k}}}$ with A_s, C_t, A_t and C_s respectively) as discussed in Section 4.1. For transfer learning, we extend (7)–(8), where $U_{C_{t|k}}$ in (12) transfers the knowledge from target to source and $U_{C_{s_{|k}}}$ in (14) transfers the knowledge from source to target, and propose the following iterative process (12)–(15)

$$U_{A_{s_{|k}}}\Sigma_{A_{s_{|k}}}V^T_{A_{s_{|k}}} = SVD(X_s U_{C_{t_{|k}}}) \tag{12}$$

$$U_{C_{s_{|k}}}\Sigma_{C_{s_{|k}}}V^T_{C_{s_{|k}}} = SVD(X_s^T U_{A_{s_{|k}}}) \tag{13}$$

$$U_{A_{t_{|k}}}\Sigma_{A_{t_{|k}}}V^T_{A_{t_{|k}}} = SVD(X_t U_{C_{s_{|k}}}) \tag{14}$$

$$U_{C_{t_{|k}}}\Sigma_{C_{t_{|k}}}V^T_{C_{t_{|k}}} = SVD(X_t^T U_{A_{t_{|k}}}) \tag{15}$$

where $U_{C_{t_{|k}}}$ is initialized as in OCM satisfying $\tan(C_{t_{|k}}, U^0_{C_{t_{|k}}}) < 1$. In our iterative process (12)–(15), four orthogonal basis cores ($U_{A_{s_{|k}}}$, $U_{C_{s_{|k}}}$, $U_{A_{t_{|k}}}$ and $U_{C_{t_{|k}}}$) are produced in each iteration where $U_{C_{s_{|k}}}$ and $U_{C_{t_{|k}}}$ gradually discover the shared subspace which is explained in Section 4.2.1.

4.2.1. Convergence of OCMTL

According to our assumption mentioned earlier, let the coefficient matrices of X_s and X_t (C_s and C_t, respectively) share a set of components (for transfer learning to be meaningful, we assume that such k shared components exist). For convergence, we also assume that among the k shared components, τ components exist in the dominant part including the most dominant components. For simplicity, let $\tau = k$. Multiplying both sides of (15) by $(C_{s_{|k}})^T$, we have

$$(C_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}}\Sigma^{i+1}_{C_{t_{|k}}}(V^{i+1}_{C_{t_{|k}}})^T = (C_{s_{|k}})^T X_t^T U^{i+1}_{A_{t_{|k}}} = (C_{s_{|k}})^T C_t \Sigma_t A_t^T U^{i+1}_{A_{t_{|k}}}$$

Since C_s and C_t share a common subspace of dimension k, then we have

$$(C_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}}\Sigma^{i+1}_{C_{t_{|k}}}(V^{i+1}_{C_{t_{|k}}})^T = \begin{bmatrix} I_k & \mathbf{0}_{k\times(n-k)} \end{bmatrix}\Sigma_t A_t^T U^{i+1}_{A_{t_{|k}}} = \Sigma_{t_{|k}} A^T_{t_{|k}} U^{i+1}_{A_{t_{|k}}} \tag{16}$$

Similarly,

$$(C^c_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}}\Sigma^{i+1}_{C_{t_{|k}}}(V^{i+1}_{C_{t_{|k}}})^T = \Sigma^c_{t_{|k}} A^{cT}_{t_{|k}} U^{i+1}_{A_{t_{|k}}} \tag{17}$$

Theorem 4.3. *The angle between the true subspace $C_{s_{|k}}$ and the iteratively constructed approximate subspace $U_{C_{t_{|k}}}$ will gradually decrease with iterations in the following form*

$$\tan\left(C_{s_{|k}}, U^{i+1}_{C_{t_{|k}}}\right) \leq \left(\frac{\lambda_{k+1}}{\lambda_k}\right)^4 \tan\left(C_{s_{|k}}, U^{i}_{C_{t_{|k}}}\right)$$

Proof. The angle between $C_{s_{|k}}$ and $U_{C_{t_{|k}}}$ after $i+1$ iterations (defined in (12)–(15)) is

$$\begin{aligned}
\tan\left(C_{s_{|k}}, U^{i+1}_{C_{t_{|k}}}\right) &= \tan\left(C_{s_{|k}}, U^{i+1}_{C_{t_{|k}}}\right) = \frac{\sigma_{max}((C^c_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}})}{\sigma_{min}((C_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}})} \\
&= \left\|(C^c_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}}\left((C_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}}\right)^{-1}\right\|_s \\
&= \left\|(C^c_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}}\Sigma^{i+1}_{C_{t_{|k}}}(V^{i+1}_{C_{t_{|k}}})^T\left((C_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}}\Sigma^{i+1}_{C_{t_{|k}}}(V^{i+1}_{C_{t_{|k}}})^T\right)^{-1}\right\|_s \\
&\leq \frac{\sigma_{max}\left((C^c_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}}\Sigma^{i+1}_{C_{t_{|k}}}(V^{i+1}_{C_{t_{|k}}})^T\right)}{\sigma_{min}\left((C_{s_{|k}})^T U^{i+1}_{C_{t_{|k}}}\Sigma^{i+1}_{C_{t_{|k}}}(V^{i+1}_{C_{t_{|k}}})^T\right)} \\
&\leq \frac{\lambda_{k+1}\sigma_{max}\left(A^{cT}_{t_{|k}} U^{i+1}_{A_{t_{|k}}}\right)}{\lambda_k \sigma_{min}\left(A^T_{t_{|k}} U^{i+1}_{A_{t_{|k}}}\right)} = \frac{\lambda_{k+1}}{\lambda_k}\tan\left(A_{t_{|k}}, U^{i+1}_{A_{t_{|k}}}\right)
\end{aligned}$$

Similarly, we can show that

$$\tan\left(A_{t_{|k}}, U^{i+1}_{A_{t_{|k}}}\right) \leq \frac{\lambda_{k+1}}{\lambda_k}\tan\left(C_{t_{|k}}, U^{i+1}_{C_{s_{|k}}}\right)$$

$$\tan\left(C_{t_{|k}}, U^{i+1}_{C_{s_{|k}}}\right) \leq \frac{\lambda_{k+1}}{\lambda_k}\tan\left(A_{s_{|k}}, U^{i+1}_{A_{s_{|k}}}\right)$$

$$\tan\left(A_{s_{|k}}, U^{i+1}_{A_{s_{|k}}}\right) \leq \frac{\lambda_{k+1}}{\lambda_k}\tan\left(C_{s_{|k}}, U^{i}_{C_{t_{|k}}}\right)$$

Therefore, $\tan\left(C_{s_{|k}}, U^{i+1}_{C_{t_{|k}}}\right) \leq \left(\frac{\lambda_{k+1}}{\lambda_k}\right)^4 \tan\left(C_{s_{|k}}, U^{i}_{C_{t_{|k}}}\right)$, which ensures the convergence of $U_{C_{t_{|k}}}$ to a subspace spanned by the columns of C_s. We can show the convergence of the other cores in a similar manner. □

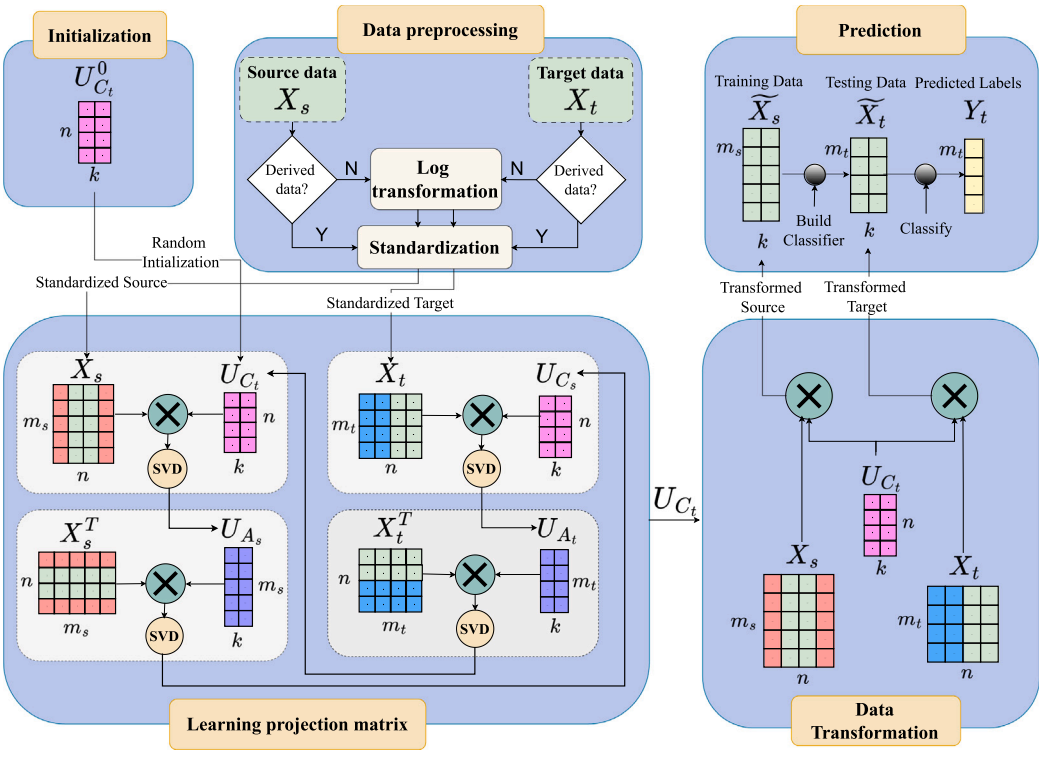


Fig. 1. Architecture of OCMTL.

According to Theorem 4.3, we can draw five important conclusions: (i) similar to OCM, the stopping criteria of OCMTL can be defined (Remark 2), (ii) the similarity between $U_{C_{s|k}}$ and $U_{C_{t|k}}$ gradually increases and they discover the shared subspace (Corollary 4.3.1), (iii) the convergence rate of OCMTL is linear (Corollary 4.3.2), (iv) OCMTL is able to capture the shared components, even if they are not in the individual dominant parts of the source and target, by identifying the dominant components of the covariance of the covariances of the source and target (Theorem 4.4, Remark 3), though the convergence requires the existence of the most dominant component of the source and target individually in the shared part, and (v) OCMTL is intrinsically regularized (Remark 4). The whole process and the architecture of OCMTL are given in Algorithm 1 and Fig. 1, respectively.

Algorithm 1 OCMTL

Input: Log transformed (optional) and standardized source data X_s and target data X_t, selected feature dimension k, $flag\_feature\_selection$
Output: Transformed source $\tilde{X}_{s_{|k}}$, transformed target $\tilde{X}_{t_{|k}}$
Initialization: $i \leftarrow 0$, Generate $U^i_{C_{t_{|k}}}$ randomly, and $U^i_{C_{s_{|k}}} \leftarrow U^i_{C_{t_{|k}}}$
repeat
  $i \leftarrow i + 1$
  Compute $U^i_{A_{s_{|k}}}$, $U^i_{C_{s_{|k}}}$, $U^i_{A_{t_{|k}}}$ and $U^i_{C_{t_{|k}}}$ according to (12), (13), (14) and (15), respectively
until $\frac{\|U^{i-1}_{C_{s_{|k}}} - U^i_{C_{s_{|k}}}\|_F}{\|U^{i-1}_{C_{s_{|k}}}\|_F} \le \epsilon$ **and** $\frac{\|U^{i-1}_{C_{t_{|k}}} - U^i_{C_{t_{|k}}}\|_F}{\|U^{i-1}_{C_{t_{|k}}}\|_F} \le \epsilon$
Compute transformed source $\tilde{X}_{s_{|k}} \leftarrow X_s U^i_{C_{t_{|k}}}$ and target $\tilde{X}_{t_{|k}} \leftarrow X_t U^i_{C_{t_{|k}}}$
if not $flag\_feature\_selection$ **then**
  return $\tilde{X}_{s_{|k}}, \tilde{X}_{t_{|k}}$
else
  return $\tilde{X}_{s_{|k}}, \tilde{X}_{t_{|k}}$, **condition_number**$(U^i_{A_{t_{|k}}})$
end if

Remark 2. It is observed from Theorem 4.3 and Corollary 4.3.1 that the angle between $U_{C_{s|k}}$ and $U_{C_{t|k}}$ can be used to identify the shared subspace between the source and target. Consequently, similar to OCM, we use both $\|U^{i+1}_{C_{s|k}} - U^i_{C_{s|k}}\|_F \le \epsilon$ and $\|U^{i+1}_{C_{t|k}} - U^i_{C_{t|k}}\|_F \le \epsilon$ as the stopping criteria.

Corollary 4.3.1. *The column spaces of $U_{C_{s|k}}$ and $U_{C_{t|k}}$ converge to the intersection of the subspaces spanned by the columns of C_s and C_t.*

Proof. Columns of the $U_{C_{s|k}}$ belong to the column space of C_s, and converge to a subspace spanned by the columns of C_t. Hence the column space of $U_{C_{s|k}}$ converges in the intersection (overlapping part) of the subspaces spanned by the columns of C_s and C_t, so does the column space of $U_{C_{t|k}}$. □

Corollary 4.3.2. *The convergence rate of OCMTL is linear.*

Proof. From Theorem 4.3, the recursive formula for convergence is

$$\tan\left(C_{s_{|k}}, U^1_{C_{t|k}}\right) \le \left(\frac{\lambda_{k+1}}{\lambda_k}\right)^{4\times 1} \tan\left(C_{s_{|k}}, U^0_{C_{t|k}}\right)$$
$$\tan\left(C_{s_{|k}}, U^2_{C_{t|k}}\right) \le \left(\frac{\lambda_{k+1}}{\lambda_k}\right)^{4\times 2} \tan\left(C_{s_{|k}}, U^0_{C_{t|k}}\right)$$
$$\vdots$$
$$\tan\left(C_{s_{|k}}, U^i_{C_{t|k}}\right) \le \left(\frac{\lambda_{k+1}}{\lambda_k}\right)^{4\times i} \tan\left(C_{s_{|k}}, U^0_{C_{t|k}}\right)$$

Since $0 < \frac{\lambda_{k+1}}{\lambda_k} < 1$, the iterative process converges with a linear rate. A similar conclusion is also asserted in the existing literature [35]. □

Theorem 4.4. *OCMTL captures the shared component(s), considering the joint impact of both the variances and the alignment between the source and target.*

Proof. For clarity, dropping the subscripts related to the dimension, (14) can be written as

$$U_{A_t}\Sigma_{A_t}V_{A_t}^T = X_tU_{C_s} \Rightarrow U_{A_t} = X_tU_{C_s}V_{A_t}\Sigma_{A_t}^{-1} \tag{18}$$

Substituting U_{C_s}, U_{A_s} and U_{C_t} from (13), (12) and (15), respectively, in (18), we have

$$U_{A_t} = (X_tX_s^TX_sX_t^T)U_{A_t}(V_{C_t}\Sigma_{C_t}^{-1}V_{A_s}\Sigma_{A_s}^{-1}V_{C_s}\Sigma_{C_s}^{-1}V_{A_t}\Sigma_{A_t}^{-1}) = SU_{A_t}Z \tag{19}$$

where $S = (X_tX_s^T)(X_tX_s^T)^T$ and is symmetric, X_s and X_t are standardized, and $Z = V_{C_t}\Sigma_{C_t}^{-1}V_{A_s}\Sigma_{A_s}^{-1}V_{C_s}\Sigma_{C_s}^{-1}V_{A_t}\Sigma_{A_t}^{-1}$. Multiplying both sides of (14) and (15) by $U_{A_t}^T$ and $U_{C_t}^T$, respectively, and assuming $U_{C_s} \approx U_{C_t}$ after convergence, we can write

$$U_{A_t}^TX_tU_{C_s} = U_{A_t}^TU_{A_t}\Sigma_{A_t}V_{A_t}^T = \Sigma_{A_t}V_{A_t}^T \tag{20}$$

$$U_{C_s}^TX_t^TU_{A_t} = U_{C_s}^TU_{C_t}\Sigma_{C_t}V_{C_t}^T \Rightarrow U_{A_t}^TX_tU_{C_s} = V_{C_t}\Sigma_{C_t}^T \tag{21}$$

From (20) and (21), we have

$$V_{C_t}\Sigma_{C_t}^T = \Sigma_{A_t}V_{A_t}^T. \tag{22}$$

Similarly,

$$V_{C_s}\Sigma_{C_s}^T = \Sigma_{A_s}V_{A_s}^T. \tag{23}$$

Since Σ_{A_t} is diagonal in (22), $V_{C_t}\Sigma_{C_t}^TV_{A_t}$ must also be diagonal. This implies that V_{C_t} and V_{A_t} act like permutation matrices. Using (22) and (23), we get Z as

$$Z = V_{C_t}\Sigma_{C_t}^{-1}V_{A_s}\Sigma_{A_s}^{-1}V_{C_s}\Sigma_{C_s}^{-1}V_{A_t}\Sigma_{A_t}^{-1} = V_{C_t}(\Sigma_{C_t}^{-1}\Sigma_{C_s}^{-1}\Sigma_{C_s}^{-1}\Sigma_{C_t}^{-1})V_{C_t}^T = D. \tag{24}$$

As long as V_{C_t} acts like a permutation matrix, D $(= \Sigma_{C_t}^{-1}\Sigma_{C_s}^{-1}\Sigma_{C_s}^{-1}\Sigma_{C_t}^{-1})$ is diagonal and $D_{i,i} = \sigma_{C_{t_i}}^{-2}\sigma_{C_{s_i}}^{-2}$ for $i = 1, \ldots, k$, (19) can be written as

$$U_{A_t} = SU_{A_t}D \;\Rightarrow\; SU_{A_t} = U_{A_t}D^{-1}. \tag{25}$$

Since S is the covariance of the covariances of the source and target, and the columns of U_{A_t} are the eigenvectors of S corresponding to the eigenvalues $D_{i,i}^{-1} = \sigma_{C_{t_i}}^2\sigma_{C_{s_i}}^2$, U_{A_t} maximizes the trace of $U_{A_t}^T(X_tX_s^T)(X_tX_s^T)^TU_{A_t}$ s.t. $\|U_{A_t}^T\| = \sqrt{k}$, where k is the number of aligned components. Therefore, U_{A_t} maximizes both the variance and alignment (in terms of correlation) of the source and target, and thereby captures the shared subspace. □

It is observed from Theorem 4.4 that the columns of U_{A_t} are the dominant components of the covariance of the covariances of the source and target, and after convergence (considering $U_{C_s} \approx U_{C_t}$), therefore U_{A_t} along with Σ_{A_t} and V_{A_t} can be found by projecting X_t onto U_{C_t}, which can be used as the testing data ($\tilde{X}_t$). Similarly, the training data ($\tilde{X}_s$) can be found. Therefore, ($\tilde{X}_s$) and ($\tilde{X}_t$) possess the aligned components (both weak and strong) that have discrimination ability.

Remark 3. Since S is symmetric, the SVD of S is

$$(X_tX_s^T)^T(X_tX_s^T) = U\Sigma U^T = UD^{-1}U^T. \tag{26}$$

Using (26), we can write

$$cov(X_tX_s^TU_{A_t}, X_tX_s^TU_{A_t}) = U_{A_t}^T(X_tX_s^T)^T(X_tX_s^T)U_{A_t} = U_{A_t}^TUD^{-1}U^TU_{A_t}. \tag{27}$$

Again, according to Theorem 4.4, (27) is maximized with the constraint $\|U_{A_t}^T\| = \sqrt{k}$, which indicates that (27) is diagonalizable only for those latents in U_{A_t} that are in the shared part. Therefore, selecting the appropriate value of k is crucial, and the proposed feature selection method determines the value of k (discussed in Section 4.3). If k is very large, it may include weakly correlated low-variance components or some uninformative components, which might reduce the discriminability of the data.

Remark 4. We perform SVD in every step of OCMTL (12)–(15). After convergence, we consider only those latents for which the condition number is lower than a certain threshold (κ) value (Section 4.3). As long as we only consider the top eigenvectors based on condition number, OCMTL is intrinsically regularized [36].

Remark 5. Since the identification of shared component(s) in OCMTL depends on covariances of the covariance of source and target, if all or the majority entries of one or more columns in either source or target are zeros, the covariance may become uninformative, and the resulting performance might degrade.

4.2.2. Rationale of OCMTL

By finding the tradeoff between discrimination ability and alignment, OCMTL is able to identify the shared components (Theorem 4.4). Thus, the cores contain the aligned components in descending order based on their dominance in the covariance of the covariances of the source and target. This ensures all strong and weak aligned components between the source and target will be part of the cores (Remark 3), if they have discrimination ability. When the ratio of the largest and the smallest singular value of the cores is small (less than $\kappa = 10$), it automatically ensures three major properties: the cores capture the dominant shared space, have better discrimination ability, and are intrinsically regularized (Remark 4). In Theorem 4.3, we assume that the most dominant components of the individual source and target are shared. This is a reasonable assumption, as excluding the top dominant components (if not shared) would discard a major portion of the variance, thereby degrading the performance even after capturing the shared ones.

When source and target are projected on $U_{C_{t_{|k}}}$, the information of the transformed source and target will possess the shared similar components and thus facilitate transfer learning. Therefore, we use $U_{C_{t_{|k}}}$ to obtain the transformed source ($\tilde{X}_{s_{|k}} = X_sU_{C_{t_{|k}}}$) and the transformed target ($\tilde{X}_{t_{|k}} = X_tU_{C_{t_{|k}}}$) without requiring any parameter. Since the columns of $U_{C_{t_{|k}}}$ are orthogonal, multicollinearity is alleviated in $\tilde{X}_{s_{|k}}$ and $\tilde{X}_{t_{|k}}$ while preserving the interactions between the features of individual source and target. For example, in CPDP, high coupling and low cohesion of a software function can indicate a higher possibility of defects. If such feature interactions are strong, OCMTL will automatically capture them in the dominant part of the cores according to Theorem 4.4.

Therefore, the best performance and convergence of OCMTL will be obtained if $\tau = k$, i.e., the shared components lie within the most dominant part of the individual source and target. In other words, case (i) when $U_{C_{t_{|k}}}$ captures the top k components of both source and target, it ensures the preservation of maximum variance in $\tilde{X}_s$ and $\tilde{X}_t$; case (ii) when the shared components are not in a consecutive order (e.g., the dominant 1st, 3rd, and 5th components of the individual source and target), OCMTL will still converge because the 1st component is shared, and it captures the remaining shared components (3rd or 5th) at the cost of reduced variance, which will degrade the classification performance (although it is the best achievable performance).

4.3. Feature selection

For preparing the training and testing data, we have to choose the number of latent features k such that the projections $X_sU_{C_{t_{|k}}}$ and $X_tU_{C_{t_{|k}}}$ are well-aligned by capturing all shared components while preserving informative structure. The feature selection is performed according to Algorithm 2, based on Remarks 3 and 4.

Algorithm 2 Feature Selection for OCMTL

Input: Standardized source data X_s, standardized target data X_t
Output: Transformed source $\tilde{X}_{s_{|k}}$, transformed target $\tilde{X}_{t_{|k}}$
Initialize $k \leftarrow 0$
repeat
  $k \leftarrow k + 1$
  $\tilde{X}_{s_{|k}}, \tilde{X}_{t_{|k}}, condition\_number \leftarrow \text{OCMTL}(X_s, X_t, k, flag\_feature\_selection)$
until $condition\_number \geq \kappa$
return $\tilde{X}_{s_{|k}}$, $\tilde{X}_{t_{|k}}$

Complexity analysis

For complexity analysis, let M $(= m_s + m_t)$, n, k and T are the total number of samples, number of features, the size of the latent space, the number of iterations, respectively. Each iteration of OCMTL consists of four steps (12)–(15). In (12), there are two types of costs: (i) matrix multiplication ($O(m_s nk)$) and (ii) SVD ($O(k^2 m_s)$). Combining these two, we get $O(m_s nk + m_s k^2)$. Therefore, considering all four steps (12)–(15), the total cost of OCMTL is $O(Mnk+k^2(M+n))$ per iteration. Considering $M > n$ and $n \gg k$, the overall major computational cost of OCMTL is $O(T(Mnk))$, where T is usually less than 10. Our closest competitors in terms of computational complexity are CORAL and JCSL, both of which are $O(Mn^2)$. Moreover, if the sample size becomes large, many methods (such as TCA, TCA+, JDA, BDA and LRCL) might require higher computational resources, as they are $O(M^3)$. In our comparison, we also used a deep learning based method (DMDA_JFR), which inherently has a much higher computational and memory complexity in comparison to others. Considering memory, OCMTL has the lowest complexity ($O(Mn)$). CORAL and JCSL also have a similar memory complexity. In contrast, other methods, such as TCA, TCA+, JDA and BDA, have much higher memory complexity ($O(M^2)$).

5. Experimental design

In this section, we provide an overview of the datasets, implementation details of OCMTL and other compared methods if necessary, performance metrics and significance test. We analyzed our experimental results in comparison with (i) state-of-the-art TL-based CPDP methods (TCA, TCA+, JDA, BDA, DMDA_JFR, CORAL and CTKCCA), (ii) state-of-the-art TL methods that can be applied to CPDP (LDA, LRCL, DALI, and JCSL) , and (iii) a baseline approach that directly trains on the source and tests on the target data and NN-Filter.

5.1. Dataset description

We use four benchmark CPDP datasets[1] in our experiment: AEEEM [37], RELINK [38], NASA [39,40], and SOFTLAB [39]. All four datasets contain common static code features related to code size (e.g., Lines of Code: Total, Blank, and Comment) and structural complexity (e.g., Cyclomatic Complexity). Besides these, each dataset contains additional features. AEEEM includes object-oriented metrics (e.g., Weighted Methods per Class, Coupling Between Objects, Response for a Class, Depth of Inheritance Tree) and process and change metrics (e.g., Number of Bugs, Number of Revisions), that introduce sparsity in many cases. Though RELINK, NASA, and SOFTLAB focus only on static code features and do not contain process or entropy-based measures like AEEEM, they contain other static features such as Branch Count, Decision Count, and Call Pairs. Details of each dataset are provided in Table 1.

Notice that the data related to all the projects are imbalanced, and the projects in both AEEEM and RELINK (written in JAVA) are more diverse compared to the projects in NASA and SOFTLAB (written in C). For example, in AEEEM, JDT and LC are projects for IDE development and search engine, respectively, which are more different compared to the projects in SOFTLAB, where all projects are related to controllers for domestic appliances. Again, there exist additional challenges in these datasets, for example, sparsity (such as AEEEM) and skewness (such as RELINK). Therefore, working with these four datasets demonstrates the capability of a method to handle diverse cases in CPDP.

5.2. Implementation details

Taking each possible pair of projects in the same dataset, we form two cross-project pairs following previous studies [10,11,29], where in each pair, one will be the source and the other will be the target, and vice-versa. For example, two projects EQ and JDT form two cross-project pairs: EQ⇒JDT (where EQ is the source and JDT is the target) and JDT⇒EQ. Following this approach, we obtain total 66 (20+20+20+6) cross-project pairs from the four datasets. We only consider homogeneous cross-project pairs and do not consider any pair where the source project is from one dataset and the target is from another.

We apply z-score normalization for all datasets. For OCMTL, we perform log transformation (when the feature data is skewed and is not derived) before standardization. In our experiment, we used the codes that are publicly available: TCA,[2] JDA,[3] BDA,[4] CORAL,[5] CTKCCA,[6] DMDA_JFR,[7] LDA,[8] and JCSL.[9] Otherwise, we implemented the methods according to the original papers (TCA+, DALI, LRCL). The implementation of OCMTL is very simple and its core part can be implemented using very few lines of MATLAB code Appendix. All experiments are executed on an Apple MacBook Air with an M1 chip (8-core CPU), 8 GB unified memory, and 256 GB storage, running macOS Sequoia version 15.6.1. For parameter values, we followed the recommended settings in the respective papers (e.g., a linear kernel and regularization parameter of 0.1 for TCA, TCA+, JDA and BDA; 10 iterations for JDA and BDA; noise probability of 0.5 and 10 layers for DMDA_JFR; 70 components in the incomplete Cholesky decomposition and a Gaussian kernel for CTKCCA), unless we state otherwise explicitly. In contrast, OCMTL does not require any parameter tuning, however, it involves a random initialization. Therefore, to minimize the

[1] https://sites.google.com/view/bda-cpdp/, accessed on October 16, 2025.

[2] https://github.com/jindongwang/transferlearning/blob/master/code/traditional/TCA, accessed on March 12, 2025.

[3] https://github.com/jindongwang/transferlearning/tree/master/code/traditional/JDA, accessed on March 12, 2025.

[4] https://github.com/jindongwang/transferlearning/blob/master/code/traditional/BDA, accessed on March 12, 2025.

[5] https://github.com/VisionLearningGroup/CORAL, accessed on March 12, 2025.

[6] https://github.com/THN-BUAA/KSETE-master/blob/master/Baselines/releaseCSKCCA-v1, accessed on March 12, 2025.

[7] https://github.com/QuanyiZou/MDMAJFR, accessed on March 12, 2025.

[8] https://github.com/zzf495/Re-implementations-of-SDA/tree/main/2021-LDA, accessed on March 12, 2025.

[9] https://users.cecs.anu.edu.au/~basura/DA_SA/, accessed on October 16, 2025.

Table 1
Dataset description.

Dataset	Project	Project description	#S	#DS	#FS	Feature description	Gran.
AEEEM (Java)	EQ	OSGi framework	324	129	61	historical defect (5) + source code (17) + source code entropy (17) + change entropy (5) + churn of source code (17)	Class
	JDT	IDE development tool	997	206			
	LC	Search engine library	691	64			
	ML	Task management system	1862	245			
	PDE	IDE development tool	1497	209			
RELINK (Java)	Apache	Web server	194	98	26	static code features (e.g., max cyclomatic complexity, average line comment, LOC)	File
	Safe	Security system	56	22			
	Zxing	Bar-code scanning library	399	118			
NASA (C)	CM1	Spacecraft instrument	327	42	37	static code features (e.g., number of lines, cyclomatic complexity, Halstead complexity, mcCabe complexity)	Func.
	MW1	Zero gravity combustion experiment	251	25			
	PC1	Flight software for satellite	696	55			
	PC3	Flight software for satellite	1073	132			
	PC4	Flight software for satellite	1276	176			
SOFTLAB (C)	ar1	Controller for white-goods	121	9	29	static code features (e.g., design complexity, Halstead count, cyclomatic complexity)	Func.
	ar3	Washing machine	63	8			
	ar4	Dish washer	107	20			
	ar5	Refrigerator	36	8			
	ar6	Controller for white-goods	101	15			

#S, #DS and #FS respectively represent the total size (number) of samples, defective samples and features.

impact of randomness on the results of OCMTL, we conduct each cross-project pair experiment twenty times and use the average score as the final result for that pair, unless stated otherwise.

For the performance comparisons in **RQ1**, **RQ2** and **RQ3** (Section 6), we use logistic regression (LR) from LIBLINEAR [41] with options "-s 0" and "-B -1", as suggested by prior defect prediction studies [10,11,13,14]. Again, we observe that setting the weight option of this LR as the "inverse of the class sample sizes of the source" improves the performance of OCMTL for higher number of features (**RQ5** in Section 6). However, with the same weight, the performance of other methods either remains similar or decreases, except CORAL, JCSL and DALI. Therefore, all the results are generated with weights for OCMTL and without weights for the other methods unless otherwise stated explicitly. We do not consider the weighted LR for CORAL, DALI and JCSL, as their original papers do not mention such weighting, nonetheless, we discuss the impact of the weights on their results in **RQ1**. In **RQ4**, we additionally compare all methods using support vector machine (SVM from LIBLINEAR) and 1-nearest neighbor (1-NN) to analyze how classifier choice affects performance.

5.3. Performance metrics

The datasets used in our experiment contain either defective or non-defective instances, so, there are four possible outcomes: true positive (TP), true negative (TN), false positive (FP) and false negative (FN). Based on these four outcomes, we can obtain the possibility of detection (pd or recall), precision and probability of false alarm (pf), which can be further used to measure the prediction performance. Recall (pd), precision and pf are defined as $\frac{TP}{TP+FN}$, $\frac{TP}{TP+FP}$ and $\frac{FP}{FP+TN}$, respectively.

All four datasets used in our experiment are imbalanced (details in Table 1). For imbalanced data, the performance is usually measured by AUC [42], F1-score (28) and G-Measure (29). Apart from them, Distance-to-heaven (d2h) (30) is also suggested in [43] for defect prediction. Since using a single metric might be misleading in some cases, we adopt these four metrics for performance comparison. The definitions of these metrics are given below:

$$\text{F1-score} = \frac{2 * precision * recall}{precision + recall} \tag{28}$$

$$\text{G-Measure} = \frac{2 * pd * (1 - pf)}{pd + (1 - pf)} \tag{29}$$

$$\text{d2h} = \sqrt{\frac{(1 - pd)^2 + (pf)^2}{2}} \tag{30}$$

AUC [42] is the area under the receiver operating characteristic (ROC) curve. The ROC curve is a two-dimensional space with pf in x-axis and pd in y-axis. AUC is independent of the threshold value for classification and thus unaffected by class imbalance. An AUC value of 0.5 represents the performance of a random predictor. Note that the range of all these metrics is from 0 to 1, where higher values of F1-score, G-Measure, AUC and lower values of d2h indicate better performance.

5.4. Significance test

To statistically analyze the performance comparisons among OCMTL and other methods, we conduct a two-step analysis using the non-parametric Friedman test followed by the Nemenyi post-hoc test [44] with a significance level, $\alpha = 0.05$. The Friedman test is conducted to determine whether there is a significant difference among the methods. The null hypothesis (H_0) of this test is that there is no significant difference among the performance of the methods. The Friedman test statistic (F_F) can be computed as:

$$F_F = \frac{(\eta - 1)\chi_F^2}{\eta(L - 1) - \chi_F^2}$$

where η is the number of cross-project pairs in a dataset, L is the number of methods being compared, and

$$\chi_F^2 = \frac{12\eta}{L(L+1)}\left[\sum_{j=1}^{L} R_j^2 - \frac{L(L+1)^2}{4}\right]$$

and R_j is the average rank of the jth method. Here, the χ_F^2 follows a χ^2 distribution with a degree of freedom, $df = L - 1$. F_F follows the F-distribution with degrees of freedom $(L - 1)$ and $(L - 1)(\eta - 1)$. We reject H_0 if the F_F value exceeds the critical value (F_α). The rejection of H_0 indicates there are significant differences in the performance of the methods and if H_0 is rejected, we proceed with the Nemenyi test to further check whether there is a significant difference between two specific methods. There is a significant difference between two particular methods if the difference of their average ranks is higher than critical difference (CD), which is given in the following equation:

$$CD = q_{\alpha,L}\sqrt{\frac{L(L+1)}{6\eta}}$$

where $q_{\alpha,L}$ is the critical value at α (0.05 in our case) significance level.

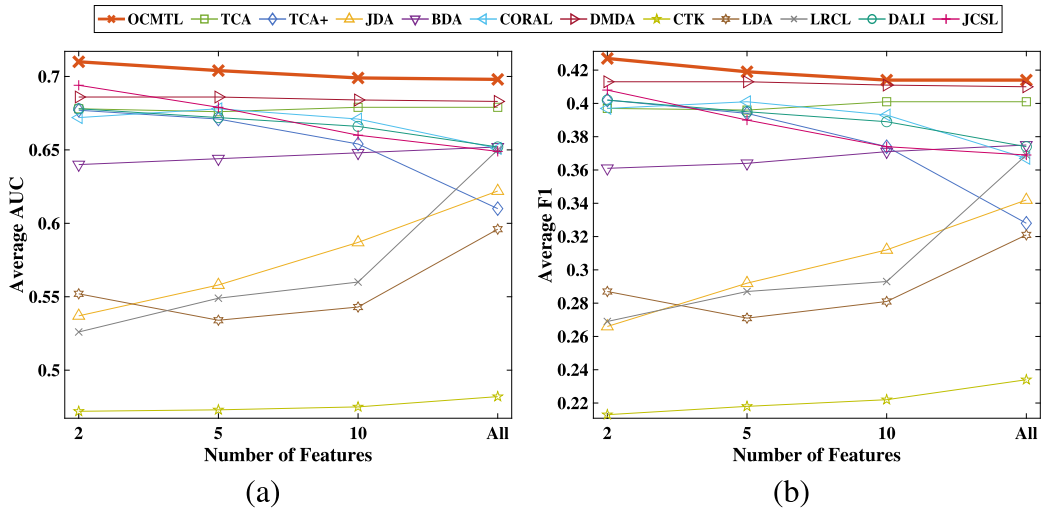


Fig. 2. Performance trend of all methods using 2, 5, 10 and all features using LR as classifier based on (a) average AUC and (b) average F1-score.

6. Results and discussion

To evaluate the performance of OCMTL in comparison with the state-of-the-art methods, we introduce the following research questions (**RQ**s). The required analysis, along with the answers to these questions, are given below. To accommodate the names of all methods in the legends of all figures and in the tables, the names of some methods are further abbreviated (e.g., baseline, NN-filter, DMDA_JFR, and CTKCCA have been abbreviated as BL, NN, DMDA, and CTK, respectively).

RQ1: What is the performance trend of OCMTL for varying numbers of features in comparison with state-of-the-art methods?

Fig. 2 represents the performance trends of all methods based on (a) average AUC and (b) average F1-score with increasing number of features (two, five, ten, and all). Each point in the Figure represents the *average-score*s (for all sixty-six cross-project pairs) of a method for a particular number of features. For example, the point presented by a cross-sign located in the top-left corner of Fig. 2(a) denotes the average AUC of 0.711 achieved by OCMTL for two features.

Notice that most of the methods show a performance-increasing tendency with increasing number of features for both AUC and F1-score. Even though the performance of DMDA_JFR, TCA, and BDA remain consistent with varying numbers of features. In contrast, OCMTL clearly outperforms all the methods and remains consistent for different numbers of features for both AUC and F1-score. An important observation is that, OCMTL exhibits comparatively better performance with a lower number of features. This is because the shared features that have better discrimination capability are usually captured within the first few columns in the latent space of OCMTL (Theorem 4.4).

CTKCCA is expected to demonstrate similar performance as its working principle is similar to OCMTL. However, expected results are not observed using their default parameter settings that include Gaussian kernel. Interestingly, the use of a linear kernel, instead, exhibits much better results. For example, with their default parameter settings and considering two features, CTKCCA achieves an average AUC of 0.452 for SOFTLAB, in contrast, with linear kernel have no parameter, it is 0.542. This also depicts how an accurate parameter-tuning (or a lack thereof) can impact the performance for a given dataset.

As mentioned in Section 5.2, using weights in the LR improves the performance of CORAL, JCSL and DALI along with OCMTL, with an increase in the number of features. In contrast, for a lower number of features, the improvement is not noticeable. With ten features, the highest improvement is observed for CORAL, where its position improves from 4th (0.671 in Fig. 2(a)) to 2nd (0.694) using AUC and from 3rd (0.393 in Fig. 2(b)) to 2nd (0.412) using F1-score. The improvements for JCSL and DALI with the same number of features using AUC are 0.660 to 0.688 and 0.666 to 0.676, respectively. A similar result has been observed for F1-score.

RQ2: How does OCMTL perform compared to other methods with the number of features selected by our feature selection algorithm?

In Fig. 2, it is observed that OCMTL with a lower number of features clearly outperforms all compared methods, regardless of the number of features used for them. Moreover, a method is considered better if it achieves higher (or similar) performance using fewer features than another method that uses more. Therefore, to answer this **RQ**, we select the number of features for OCMTL based on the proposed feature selection method (mentioned within the parentheses in the Tables 2–5) and use the same number of features for the other methods for fair comparison.

To answer **RQ2**, we first compare the performance in terms of three different criteria: *win/tie/loss*, *overall-winner*, and *average-score* based on AUC, presented in Tables 2–5. We then compare the performance based on F1-score using radar plots, demonstrated in Fig. 3. Here, *win/tie/loss* means the number of cross-project pairs for which OCMTL performs better/equally-well/worse than a particular method. The score of a method that performs the best among all for a cross-project pair is marked in bold, and this method is defined as the *overall-winner* for that specific pair. The *average-score* for a metric is the mean of all cross-project pairs' scores in a dataset obtained for a particular method.

Tables 2–5 demonstrate that OCMTL performs the best among all compared methods in terms of the three aforementioned criteria (except the number of *overall-winner*s in SOFTLAB). For example, it achieves a higher number of *overall-winner* in three out of the four datasets. Although the number of *overall-winner* of OCMTL in SOFTLAB is lower than TCA+, CORAL and JCSL, its performance in terms of both *average-score* and *win/tie/loss* is clearly better than that of all methods. A similar performance is also observed in the other datasets, except in AEEEM, where DMDA_JFR (uses deep feature representation learning) achieves competitive results to OCMTL in terms of both *average-score* and *win/tie/loss*.

TL methods are expected to perform better on datasets with similar projects (NASA, SOFTLAB) and comparatively worse on more diverse projects (AEEEM, RELINK). However, this pattern is not clearly seen for many methods, likely because they depend on parameter settings that are hard to tune. In contrast, this trend is evident for JCSL and CORAL, which perform better in NASA (average AUC of 0.675 and 0.699, respectively) and SOFTLAB (0.691 and 0.707, respectively) compared to AEEEM (0.664 and 0.686, respectively) and RELINK (0.657 and 0.653, respectively), as they have few or no parameters. In contrast,

Table 2
AUC scores of all methods on all cross-project pairs in AEEEM using LR.

Pair	OCMTL	BL	NN	TCA	TCA+	JDA	BDA	CORAL	DMDA	CTK	LDA	LRCL	DALI	JCSL
EQ⇒JDT	**0.751** (2)	0.590	0.634	0.661	0.723	0.524	0.749	0.725	0.749	0.498	0.573	0.513	0.680	0.711
EQ⇒LC	0.706 (3)	0.626	0.622	0.660	0.624	0.440	0.664	0.624	0.724	0.460	0.475	0.582	0.708	**0.740**
EQ⇒ML	0.618 (2)	0.464	0.470	0.637	0.653	0.615	**0.665**	0.650	0.652	0.487	0.501	0.526	0.592	0.614
EQ⇒PDE	0.666 (2)	0.627	0.648	0.647	0.670	0.633	0.634	0.666	**0.682**	0.485	0.504	0.496	0.644	0.674
JDT⇒EQ	**0.717** (2)	0.578	0.672	0.690	0.659	0.344	0.685	0.650	0.678	0.433	0.419	0.589	0.658	0.678
JDT⇒LC	**0.727** (2)	0.579	0.599	0.717	0.630	0.447	0.654	0.654	0.719	0.464	0.462	0.550	0.716	0.699
JDT⇒ML	0.648 (3)	0.582	0.565	0.635	**0.666**	0.473	0.635	0.659	0.652	0.486	0.489	0.512	0.648	0.640
JDT⇒PDE	0.679 (2)	0.581	0.690	0.668	0.669	0.428	0.652	0.667	0.679	0.485	0.389	0.523	**0.697**	0.689
LC⇒EQ	**0.707** (2)	0.653	0.594	0.652	0.638	0.633	**0.707**	0.654	0.678	0.500	0.675	0.578	0.671	0.678
LC⇒JDT	0.755 (2)	0.683	0.648	0.750	0.709	0.550	**0.757**	0.722	0.747	0.498	0.538	0.528	0.739	0.755
LC⇒ML	0.637 (3)	0.481	0.499	0.618	0.532	0.548	0.626	0.650	**0.655**	0.487	0.617	0.501	0.644	0.630
LC⇒PDE	**0.682** (2)	0.608	0.642	0.669	0.658	0.449	0.606	0.670	**0.682**	0.485	0.669	0.529	0.665	**0.682**
ML⇒EQ	**0.709** (2)	0.554	0.498	0.671	0.636	0.630	0.528	0.635	0.678	0.498	0.647	0.331	0.662	0.687
ML⇒JDT	0.713 (2)	0.633	0.632	**0.758**	0.649	0.633	0.678	0.713	0.745	0.497	0.370	0.555	0.750	0.724
ML⇒LC	0.707 (2)	0.519	0.509	0.700	0.654	0.509	**0.734**	0.639	0.707	0.460	0.581	0.501	0.717	0.693
ML⇒PDE	**0.680** (2)	0.579	0.619	0.660	0.652	0.562	0.553	0.656	0.678	0.485	0.568	0.483	0.666	**0.680**
PDE⇒EQ	**0.694** (2)	0.626	0.638	0.653	0.654	0.634	0.616	0.653	0.681	0.433	0.399	0.385	0.665	0.687
PDE⇒JDT	0.750 (2)	0.653	0.701	**0.767**	0.728	0.405	0.744	0.721	0.750	0.495	0.424	0.496	0.742	0.743
PDE⇒LC	0.725 (2)	0.628	0.632	0.725	0.625	0.432	0.584	0.624	0.718	0.464	0.398	0.507	**0.727**	0.689
PDE⇒ML	0.629 (2)	0.575	0.586	0.654	**0.655**	0.549	0.475	0.648	0.651	0.481	0.516	0.548	0.639	0.628
win/tie/loss		20/0/0	19/0/1	15/1/4	16/0/4	20/0/0	16/1/3	14/2/4	9/4/7	20/0/0	20/0/0	20/0/0	12/1/7	13/3/4
Avg AUC	**0.695**	0.591	0.605	0.680	0.654	0.522	0.647	0.664	**0.695**	0.479	0.511	0.512	0.682	0.686

Table 3
AUC scores of all methods on all cross-project pairs in RELINK using LR.

Pair	OCMTL	BL	NN	TCA	TCA+	JDA	BDA	CORAL	DMDA	CTK	LDA	LRCL	DALI	JCSL
Apache⇒Safe	**0.757** (3)	0.699	0.672	0.692	0.721	0.544	0.699	0.721	0.700	0.426	0.708	0.501	0.693	0.684
Apache⇒Zxing	**0.648** (2)	0.566	0.547	0.609	0.581	0.534	0.599	0.579	0.592	0.459	0.423	0.607	0.589	0.595
Safe⇒Apache	**0.742** (2)	0.553	0.541	0.682	0.682	0.624	0.666	0.682	0.651	0.446	0.677	0.598	0.687	0.682
Safe⇒Zxing	**0.640** (2)	0.501	0.569	0.610	0.582	0.592	0.609	0.578	0.594	0.473	0.493	0.566	0.572	0.602
Zxing⇒Apache	**0.752** (2)	0.664	0.656	0.650	0.671	0.480	0.640	0.666	0.656	0.467	0.640	0.505	0.687	0.677
Zxing⇒Safe	**0.749** (2)	0.640	0.582	0.684	0.744	0.617	0.715	0.715	0.700	0.545	0.721	0.648	0.669	0.677
win/tie/loss		6/0/0	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0
Avg AUC	**0.715**	0.604	0.594	0.655	0.664	0.565	0.655	0.657	0.649	0.469	0.610	0.571	0.650	0.653

Table 4
AUC scores of all methods on all cross-project pairs in NASA using LR.

Pair	OCMTL	BL	NN	TCA	TCA+	JDA	BDA	CORAL	DMDA	CTK	LDA	LRCL	DALI	JCSL
CM1⇒MW1	0.699 (3)	0.520	0.510	0.647	0.727	0.520	0.616	**0.734**	0.709	0.477	0.414	0.470	0.696	0.729
CM1⇒PC1	**0.787** (2)	0.507	0.701	0.734	0.668	0.483	0.749	0.670	0.665	0.519	0.590	0.483	0.661	0.766
CM1⇒PC3	**0.746** (2)	0.506	0.664	0.687	0.604	0.503	0.733	0.663	0.685	0.499	0.454	0.421	0.708	0.719
CM1⇒PC4	**0.709** (4)	0.473	0.686	0.617	0.673	0.383	0.664	0.657	0.644	0.496	**0.709**	0.442	0.595	0.608
MW1⇒CM1	0.675 (2)	0.576	0.579	**0.681**	0.609	0.530	0.430	0.645	0.646	0.431	0.488	0.617	0.617	0.662
MW1⇒PC1	0.724 (2)	0.511	0.602	0.559	**0.725**	0.502	0.604	0.721	0.665	0.517	0.553	0.514	0.643	0.713
MW1⇒PC3	0.717 (2)	0.523	0.603	0.682	**0.757**	0.492	0.530	0.720	0.679	0.500	0.377	0.525	0.687	0.729
MW1⇒PC4	0.657 (2)	0.476	0.574	0.652	0.616	0.451	0.550	0.632	0.636	0.499	**0.666**	0.345	0.627	0.632
PC1⇒CM1	**0.714** (2)	0.566	0.655	0.644	0.669	0.475	0.609	0.679	0.640	0.435	0.486	0.529	0.624	0.700
PC1⇒MW1	0.706 (2)	0.516	0.577	0.600	0.689	0.692	0.469	0.689	**0.709**	0.477	0.645	0.639	0.700	0.703
PC1⇒PC3	**0.742** (2)	0.523	0.711	0.647	0.712	0.487	0.736	0.710	0.678	0.500	0.573	0.449	0.708	0.733
PC1⇒PC4	0.715 (3)	0.544	**0.722**	0.644	0.654	0.465	0.664	0.654	0.643	0.496	0.447	0.424	0.636	0.620
PC3⇒CM1	**0.706** (2)	0.558	0.583	0.660	0.563	0.528	0.491	0.582	0.644	0.435	0.487	0.494	0.617	0.671
PC3⇒MW1	0.699 (2)	0.511	**0.734**	0.720	0.705	0.574	0.458	0.674	0.709	0.477	0.711	0.532	0.725	0.712
PC3⇒PC1	**0.789** (2)	0.584	0.774	0.660	0.755	0.581	0.670	0.717	0.666	0.517	0.501	0.442	0.660	0.758
PC3⇒PC4	**0.716** (3)	0.506	0.705	0.671	0.623	0.432	0.488	0.619	0.639	0.496	0.573	0.571	0.651	0.636
PC4⇒CM1	**0.689** (2)	0.459	0.540	0.641	0.642	0.511	0.656	0.642	0.642	0.435	0.654	0.474	0.613	0.675
PC4⇒MW1	0.703 (2)	0.489	0.412	0.702	**0.731**	0.689	0.714	0.711	0.711	0.477	0.667	0.472	0.709	0.720
PC4⇒PC1	**0.782** (2)	0.526	0.673	0.710	0.677	0.522	0.500	0.676	0.674	0.517	0.581	0.462	0.733	0.770
PC4⇒PC3	0.739 (2)	0.498	0.609	0.724	0.657	0.533	**0.750**	0.700	0.688	0.500	0.632	0.419	0.675	0.722
win/tie/loss		20/0/0	18/0/2	18/0/2	15/0/5	20/0/0	18/0/2	17/0/3	16/0/4	20/0/0	17/0/3	20/0/0	18/0/2	16/0/4
Avg AUC	**0.721**	0.519	0.631	0.664	0.673	0.518	0.604	0.675	0.669	0.485	0.560	0.486	0.664	0.699

OCMTL performs well in both similar (average AUC of 0.721 in NASA and 0.717 in SOFTLAB) and diverse datasets (average AUC of 0.695 in AEEEM and 0.715 in RELINK) compared to others, as it automatically optimizes alignment and variance.

The performance comparisons based on F1-scores are presented using radar plots in Fig. 3, where the shaded region in a particular plot represents the maximum performance of OCMTL for all cross-project pairs, each axis of a plot represents a cross-project pair of a dataset, and the points on the axes indicate the F1-scores of different methods for that pair. For example, the 20 axes in Fig. 3(a) represent the 20 different cross-project pairs of AEEEM. A closer look at the plots reveals that the OCMTL outperforms most of the methods in all datasets based on the previously mentioned three different criteria.

Table 5
AUC scores of all methods on all cross-project pairs in SOFTLAB using LR.

Pair	OCMTL	BL	NN	TCA	TCA+	JDA	BDA	CORAL	DMDA	CTK	LDA	LRCL	DALI	JCSL
ar1⇒ar3	0.713 (2)	0.442	0.576	0.702	0.739	0.666	0.747	0.739	0.720	0.473	0.684	0.666	**0.766**	0.729
ar1⇒ar4	0.719 (2)	0.630	0.676	0.718	**0.808**	0.580	0.687	0.758	0.749	0.503	0.658	0.562	0.691	0.700
ar1⇒ar5	0.855 (2)	0.813	0.429	0.830	0.848	0.741	0.455	0.830	0.848	0.482	0.875	0.214	0.830	**0.892**
ar1⇒ar6	0.623 (2)	0.488	0.602	0.520	0.571	0.546	0.563	0.571	0.643	0.343	**0.695**	0.240	0.633	0.586
ar3⇒ar1	0.576 (2)	0.500	0.343	0.541	**0.637**	0.526	0.586	0.530	0.581	0.403	0.562	0.448	0.562	0.576
ar3⇒ar4	0.772 (3)	0.514	0.386	0.727	**0.814**	0.382	0.674	0.764	0.749	0.478	0.480	0.589	0.714	0.771
ar3⇒ar5	0.850 (2)	0.625	0.616	0.830	0.848	0.571	0.679	0.830	0.848	0.464	0.375	0.643	**0.866**	0.848
ar3⇒ar6	**0.641** (2)	0.567	0.578	0.616	0.598	0.465	0.621	0.616	0.610	0.357	0.492	0.562	0.621	0.638
ar4⇒ar1	0.619 (2)	0.482	**0.730**	0.628	0.637	0.539	0.577	0.572	0.581	0.501	0.488	0.615	0.562	0.521
ar4⇒ar3	0.711 (2)	0.695	0.610	0.730	**0.739**	0.657	0.720	**0.739**	0.720	0.491	0.503	0.702	0.730	0.729
ar4⇒ar5	0.813 (2)	0.777	0.661	0.830	0.830	0.723	0.830	0.830	**0.848**	0.464	0.321	0.813	0.830	0.830
ar4⇒ar6	0.626 (2)	0.504	0.597	0.621	0.604	0.592	0.626	0.582	**0.643**	0.326	0.369	0.479	0.633	0.570
ar5⇒ar1	0.631 (2)	0.529	0.666	0.521	0.582	0.521	0.553	0.539	0.581	0.452	0.699	**0.728**	0.641	0.521
ar5⇒ar3	0.711 (2)	0.730	0.710	0.711	**0.739**	0.558	0.702	**0.739**	0.720	0.620	0.432	0.611	0.711	0.720
ar5⇒ar4	0.768 (2)	0.718	0.766	0.737	0.733	0.485	0.710	0.733	0.749	0.461	0.622	0.691	0.768	**0.777**
ar5⇒ar6	0.637 (2)	0.520	0.495	0.637	0.565	0.479	0.627	**0.655**	0.610	0.329	0.481	0.422	0.621	0.632
ar6⇒ar1	0.643 (2)	0.646	0.555	0.590	0.529	0.688	**0.739**	0.524	0.581	0.435	0.370	0.684	0.512	0.654
ar6⇒ar3	0.750 (3)	0.728	0.640	0.730	0.739	0.693	0.720	0.739	0.720	0.444	0.720	0.666	0.739	**0.773**
ar6⇒ar4	0.764 (2)	0.587	0.589	0.766	0.722	0.541	0.641	0.697	0.749	0.503	0.657	0.728	0.716	**0.779**
ar6⇒ar5	**0.911** (2)	0.821	0.714	0.848	0.848	0.696	**0.911**	0.830	0.848	0.500	0.795	0.741	0.795	0.892
win/tie/loss		18/0/2	18/0/2	14/2/4	12/0/8	19/0/1	14/1/5	14/0/6	12/0/8	20/0/0	17/0/3	17/1/2	11/2/7	10/1/9
Avg AUC	**0.717**	0.616	0.597	0.692	0.706	0.582	0.668	0.691	0.705	0.451	0.564	0.590	0.697	0.707

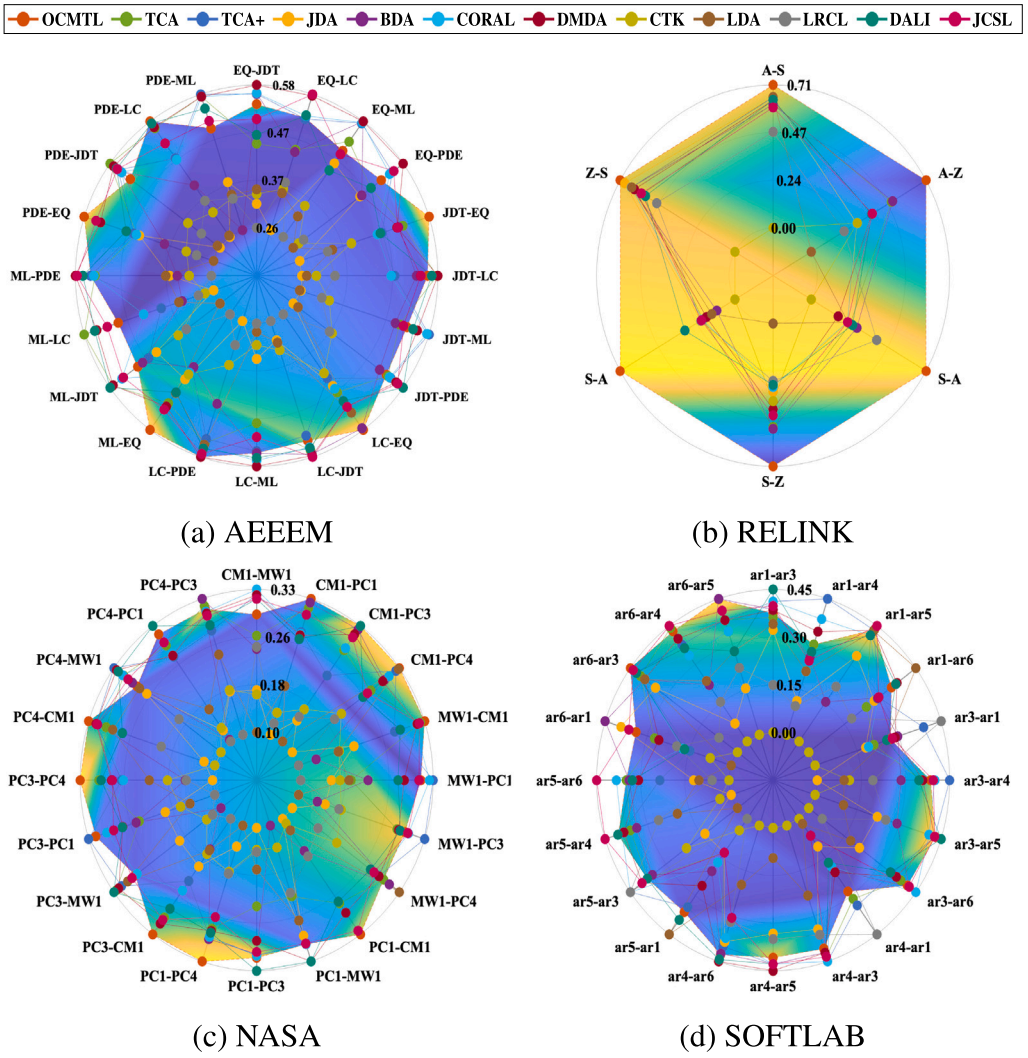


Fig. 3. Performance comparison of OCMTL with other methods based on F1-score using LR.

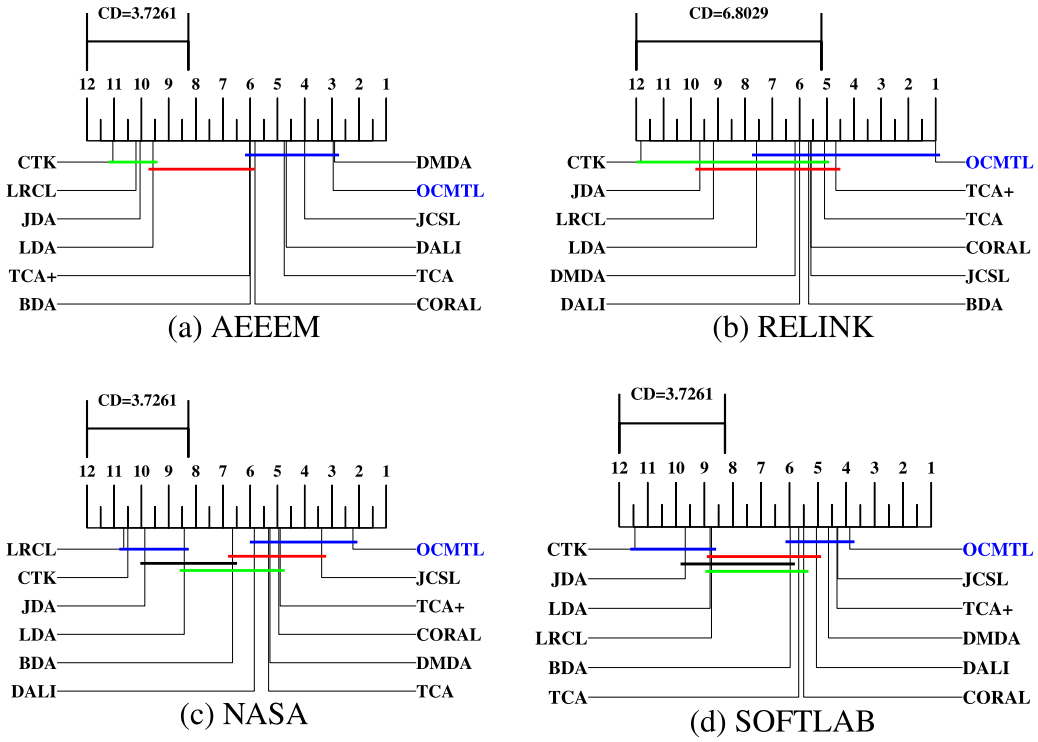


Fig. 4. The results of Nemenyi's post-hoc test for all methods based on AUC in four datasets: (a) AEEEM, (b) RELINK, (c) NASA, and (d) SOFTLAB using LR.

The performance in terms of F1 is very similar to AUC. For example, in RELINK and NASA, OCMTL is clearly better than the others in terms of *overall-winner*, *win/tie/loss* and *average-score*. In SOFTLAB, even though the number of *overall-winner* is lower than many methods, similar to AUC, its *average-score* and *win/tie/loss* is better than all other methods. However, there are a few exceptions compared to AUC. In SOFTLAB, the average F1-score of TCA+ (0.430) is very close to that of OCMTL (0.432), whereas, OCMTL (0.717) clearly outperformed TCA+ (0.706) in terms of average AUC. Moreover, in AEEEM, the performance of DMDA_JFR (uses deep feature representation learning) in terms of average F1-score and *win/tie/loss* is better than that of OCMTL, whereas they are competitive in terms of AUC. In addition, though OCMTL is clearly better than CORAL, DALI and JCSL in terms of *win/tie/loss* based on AUC, their performances are similar to OCMTL based on F1-score.

Note that the performance comparison discussed above also demonstrates the strength of the proposed feature selection method. The performance of OCMTL with the selected number of features is similar to the best corresponding performance of OCMTL observed in Fig. 2. Along with OCMTL, other state-of-the-art methods also perform reasonably well with the selected number of features in most of the cases, which shows the effectiveness of the feature selection method.

Analyzing Tables 2–5, it is observed that the traditional methods are much worse than most of the transfer learning methods. A similar observation can be made for the F1-score as well (not shown in Fig. 3). However, the performance of a few transfer learning methods (LDA, LRCL, JDA and CTKCCA) are not satisfactory, which is not expected. This is mainly due to the lack of appropriate parameter tuning for these datasets, though we use the exact parameter settings suggested in the respective papers. We believe that proper tuning of these parameters (which is hard to determine when the dataset changes) will improve their performance, and this motivates us to develop a parameter-free simple transfer learning method like CORAL.

RQ3: Does OCMTL outperform the mentioned state-of-the-art methods significantly?

To assess the significance of OCMTL's performance, we apply the Friedman test followed by the Nemenyi post-hoc test. We rank each of the methods for each of the cross-project pairs based on AUC and F1-score separately, and calculate the average rank for each of the methods for a particular dataset based on AUC and F1-score separately. To minimize the impact of randomness for OCMTL, we run the experiments twenty times for each pair and conduct the Friedman test for each run. Based on the process described in Section 5.4, we observe that the F_F value is greater than F_α for all runs in each of the datasets, which indicates that the ranks of the methods are significantly different.

We evaluate twenty Nemenyi tests corresponding to the twenty runs of OCMTL for each dataset. In all cases, OCMTL consistently achieves top or near-top ranks. Specifically, OCMTL ranks first for all cases in the RELINK and NASA datasets for both AUC and F1-score. In AEEEM and SOFTLAB, OCMTL ranks either first or second in terms of AUC (first, second or third in terms of F1-score). We illustrate the results of the Nemenyi test for one representative case for each dataset in terms of AUC in Fig. 4, where methods connected by the same horizontal line do not differ significantly. In this Figure, there are three groups for AEEEM based on the connected methods, where lower ranks (right side of the axis) indicate better performance. These results demonstrate that OCMTL performs consistently well across all datasets with the selected number of features.

RQ4: What is the impact of different classifiers on the performance of OCMTL in comparison to other methods?

For this **RQ**, we compute the average AUC, F1-score, G-Measure and d2h of OCMTL and other methods with the selected number of features using LR, SVM and 1-NN classifiers for all datasets, and present the performance comparison using radar plots in Figs. 5–7. In each radar plot, the axes represent the datasets, and the points on each axis indicate the average metric scores of the methods for the dataset represented by the axis. The shaded area in a radar plot represents the overall performance of OCMTL in all datasets. The larger the shaded area, the better the performance of OCMTL for AUC, F1-score and G-Measure. In contrast, a smaller shaded area represents a better performance for d2h.

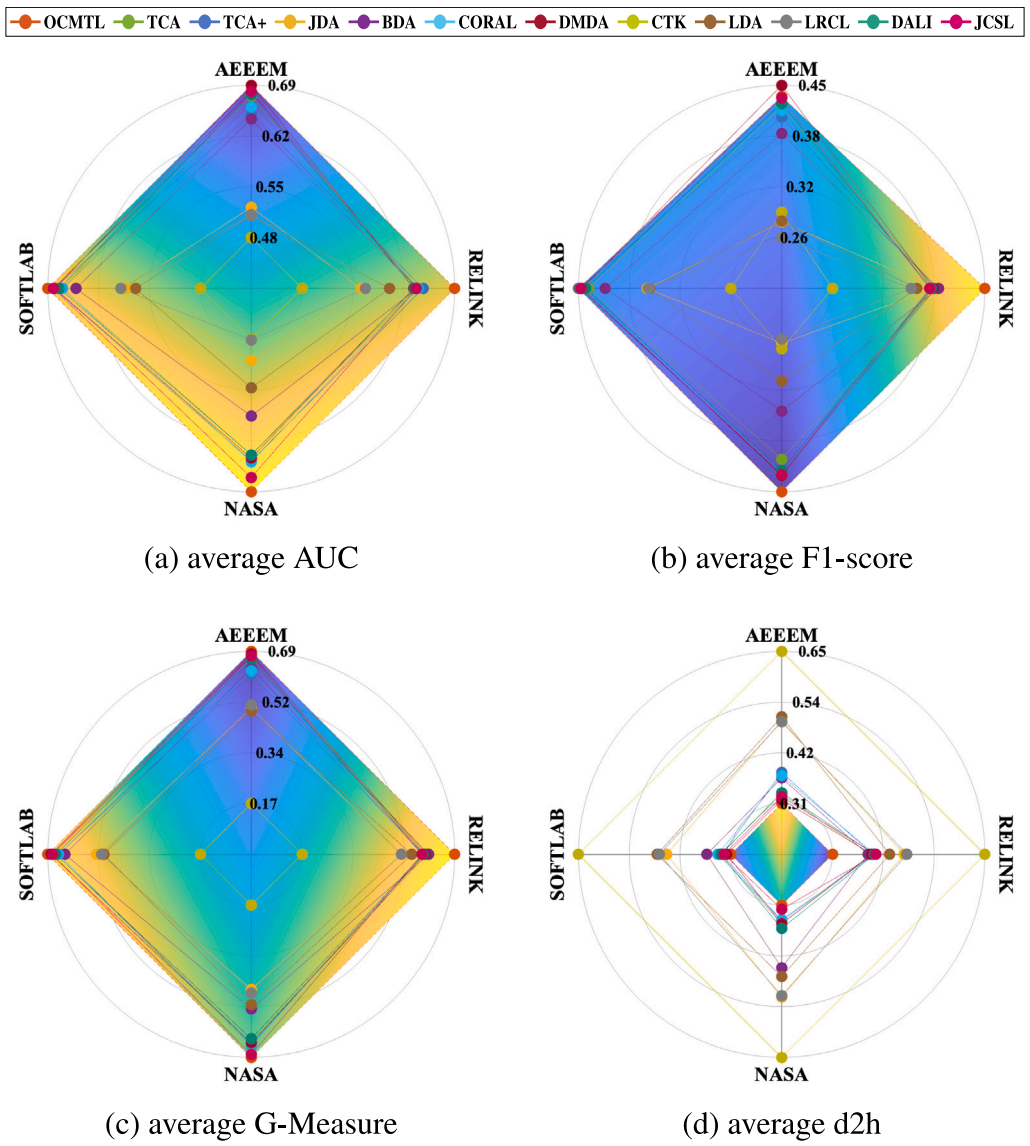


Fig. 5. Performance comparison of OCMTL and other methods on four datasets using LR.

Analyzing Figs. 5–6, it is observed that the performance behavior of OCMTL with SVM is similar to that with LR. That is, it performs better compared to others for three datasets, namely RELINK, NASA and SOFTLAB in terms of average AUC and F1-score. Moreover, in AEEEM, DMDA_JFR performs equivalently with OCMTL in terms of average AUC (0.696 for both), and better in terms of average F1-score (0.430 for OCMTL and 0.447 for DMDA_JFR), as observed with LR. In terms of average G-Measure and d2h, OCMTL with SVM (Fig. 6) and LR (Fig. 5) also show a similar performance behavior.

We have experimentally found that our feature selection method selects those components that always include the most dominant components of the individual source and target (which guarantees the convergence of OCMTL according to Theorem 4.3), along with a few other components by the tradeoff between variance and alignment (which assures better transfer learning and classification according to Theorem 4.4 and Remark 3).

The superiority of OCMTL is understandable based on the explanation given in Section 4.2.2. That is, OCMTL achieves the best performance in RELINK and NASA, because $\tau = k$ (case (i), Section 4.2.2) holds for most of the cross-project pairs in these datasets (experimentally found). In SOFTLAB, many pairs satisfy this condition, and there are cases where this condition does not hold (case (ii), Section 4.2.2), however, we get a higher *average-score* based on Theorem 4.4. Similar to SOFTLAB, both the cases also appear for AEEEM, however, due to the presence of sparsity Remark 5, a comparatively lower average performance is observed in AEEEM than the other three datasets.

A slightly different performance behavior of OCMTL is observed in the case of 1-NN compared to SVM and LR, which is understandable as 1-NN operates based on Euclidean distance, and struggles as there exists a large gap among the values of the latents in many cases. From Fig. 7, it can be seen that OCMTL performs close to the best performing methods in NASA and SOFTLAB (DMDA_JFR and CORAL, respectively) considering average AUC, and performs lower than DMDA_JFR and DALI in NASA in terms of average F1-score. In contrast, for the AEEEM and RELINK datasets, OCMTL achieves the best performance based on both average AUC and F1-score. It is also observed from Fig. 7 that the performance behavior of OCMTL in terms of average G-Measure and d2h is consistent with its performance in terms of average AUC and F1-score. The only exception is observed in RELINK, where BDA achieves an unusually high average G-Measure that does not align with its performance in other cases. We plan to investigate this further in future work.

RQ5: What is the impact of the different components of OCMTL on the performance?

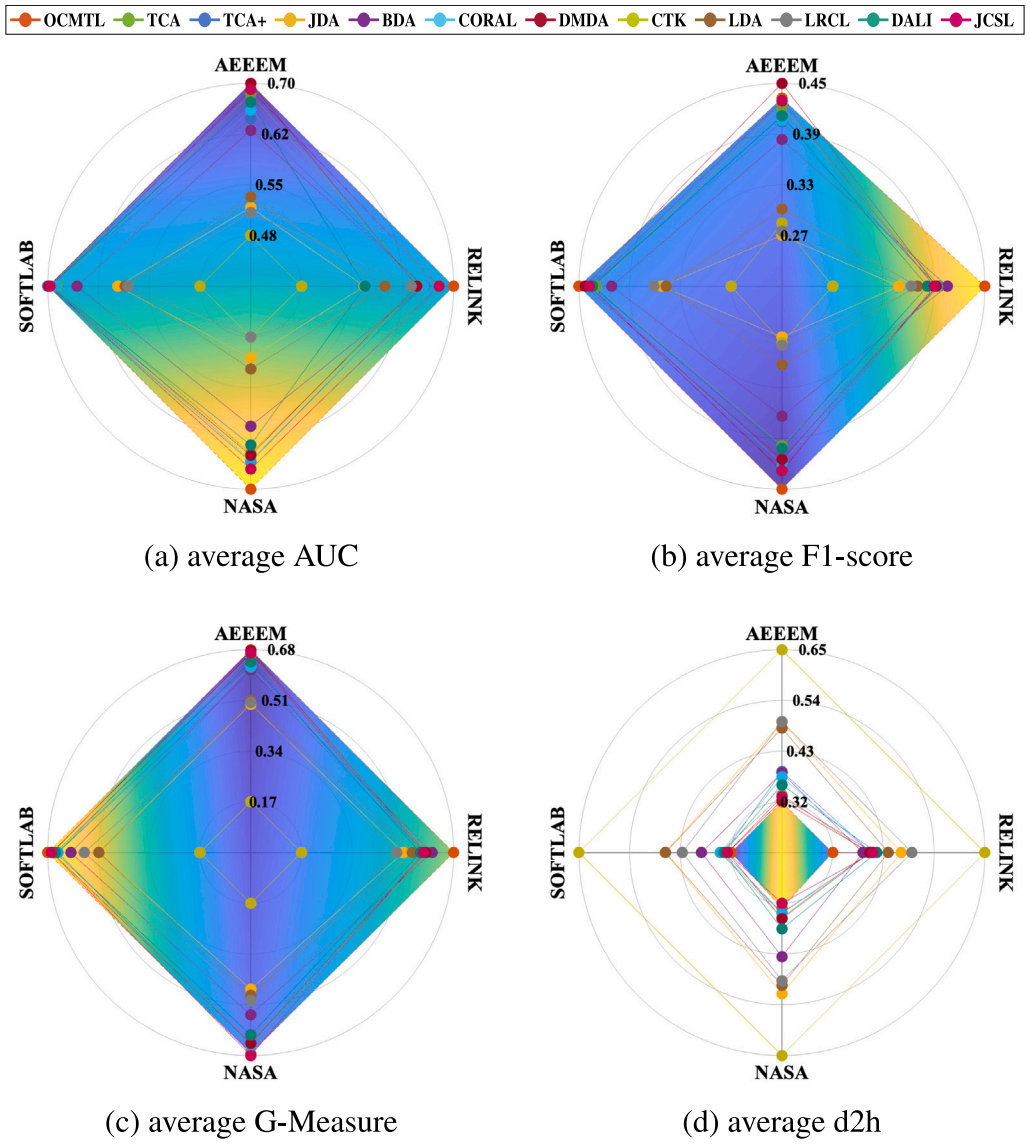


Fig. 6. Performance comparison of OCMTL and other methods on four datasets using SVM.

The impacts of different components of OCMTL, namely log transformation, imbalance weights in the classifier, selected feature dimensions, and random initialization, and the impact of different classifiers, are discussed here.

Impact of log transformation and imbalance weights in the classifier

The bargraph in Fig. 8 reports the average AUC of the basic (B) four steps of OCMTL (12)–(15) using LR (B+LR), basic four steps with log transformation (Log+B+LR), and basic four steps with weights in the LR classifier (B+LR_CW) for all datasets with 2, 5, 10 and all features. It is observed that the log transformation (Log+B+LR) achieves a clear improvement in many cases. Moreover, the use of class weights (B+LR_CW) achieves a noticeable improvement for higher number of features in most of the cases, except for SOFTLAB with all features. This gives us an indication that the incorporation of both log transformation and imbalance weights in the classifier will further improve the performance of OCMTL.

Impact of number of features and random initialization

Based on the aforementioned analysis, we now compute the performance of OCMTL with both log transformation and imbalance weights for all datasets with different numbers of features in terms of (a) average AUC and (b) average F1-score. For each dataset and number of features, the *average-score* is calculated twenty times and the average of them is reported as the final score for that case. We present the results in Fig. 9, where each scatter point denotes the *average-score* of OCMTL for a particular number of selected features. It is observed that the best results are obtained with fewer features (e.g., two to five), and the performance does not degrade much as the number of features increases. This demonstrates that our method is not highly sensitive to the number of features.

Fig. 9 also illustrates the impact of random initialization, where the 95% confidence intervals of the average scores are displayed near the scatter points. It can be seen that the intervals are very small in all cases, which confirms that random initialization has minimal impact on the results of OCMTL.

Impact of different classifiers

The performance of OCMTL in terms of average AUC, F1-score, G-Measure and d2h is analyzed for three different classifiers, namely LR, SVM and 1-NN. From Table 6, we can observe that OCMTL with the LR classifier generally performs better in most of the cases, especially with

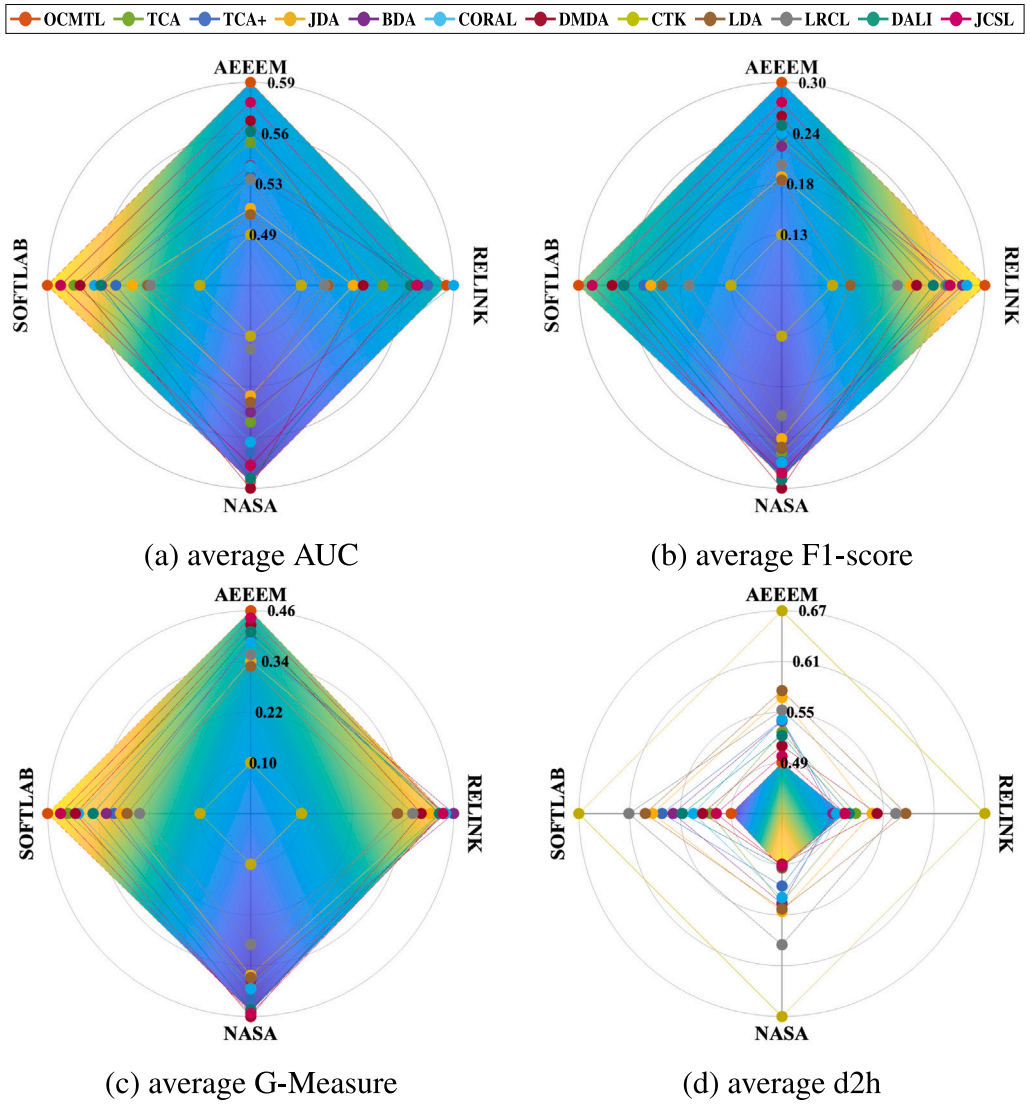


Fig. 7. Performance comparison of OCMTL and other methods on four datasets using 1-NN.

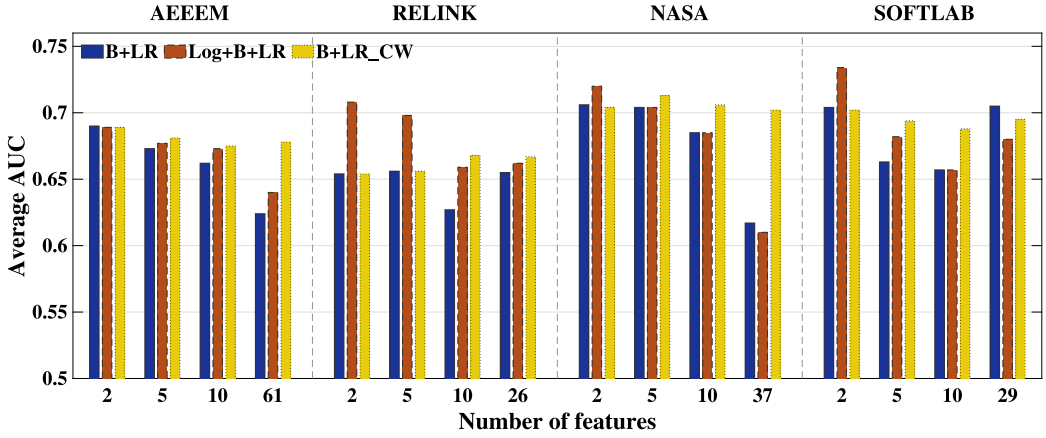
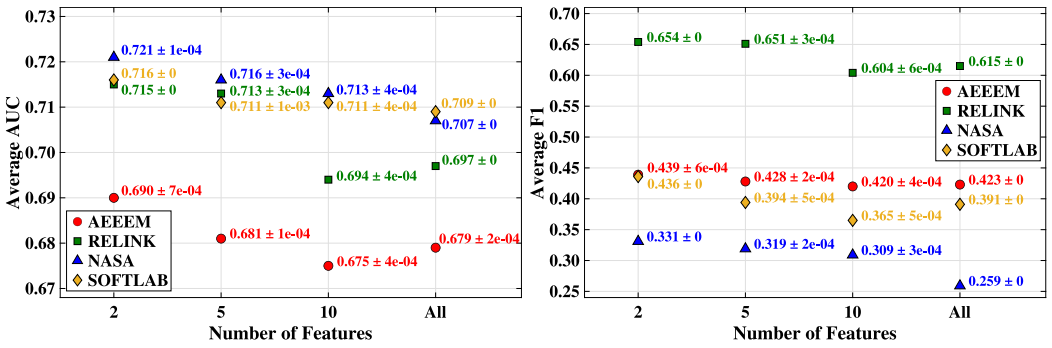


Fig. 8. Performance of OCMTL using logistic regression without log and weights (B+LR), with log (Log+B+LR), and with weights (B+LR_CW) for all datasets with varying feature dimensions.



(a) Average AUC (b) Average F1-score

Fig. 9. Performance of OCMTL across all datasets with varying number of features using LR.

Table 6
Performance of OCMTL with different classifiers for all datasets in terms of four metrics.

	AUC			F1-score			G-Measure			d2h		
	LR	SVM	1-NN	LR	SVM	1-NN	LR	SVM	1-NN	LR	SVM	1-NN
AEEEM	0.695	**0.696**	0.591	**0.434**	0.430	0.302	**0.687**	0.683	0.457	**0.313**	0.317	0.492
RELINK	**0.715**	0.706	0.604	**0.661**	0.649	0.499	**0.715**	0.706	0.531	**0.285**	0.294	0.451
NASA	**0.721**	0.719	0.551	0.341	**0.351**	0.192	**0.705**	0.689	0.325	**0.300**	0.319	0.567
SOFTLAB	**0.717**	0.706	0.650	0.432	**0.437**	0.391	0.713	**0.717**	0.562	**0.288**	0.289	0.418

weights for a higher number of features. Therefore, LR with weight is the most appropriate classifier for OCMTL.

7. Threats to validity

To avoid any inconsistency in the implementations, we use the source codes of the compared methods that are publicly available, otherwise, we try our best to implement the methods by strictly following the recommended details in the original papers. We adopt the parameter values specified in the corresponding papers, however, there is a chance of variation in results, as most of these methods are parameter-sensitive. We also experience similar cases where a few transfer learning methods (need parameter tuning) perform worse compared to even the traditional methods.

We experiment on four benchmark CPDP datasets developed in JAVA or C languages and consisting of total eighteen projects with total sixty six cross-project pairs. We believe the generalization ability of OCMTL is demonstrated in the results as these datasets are diverse in different aspects such as size, feature set, and defect rate. However, experimentation on more defect datasets will further ensure the generalization ability of OCMTL. To select the size of features, we use the condition number $\kappa = 10^p$. Our finding is that OCMTL is not sensitive to κ. Though setting $p = 0.4$ to 0.8 works fine for these four datasets, slight adjustments might need to be made for other datasets.

We carefully choose four performance metrics to evaluate the performance of all methods. We believe these metrics are robust to represent the performance of a method even for imbalanced datasets. Moreover, we use 1-NN as a classifier to highlight the comparative strengths of the methods. Other stronger classifiers such as XGBoost will surely improve the results of all methods. We also conduct the Friedman test and the Nemenyi test to check the significant differences between OCMTL and the other methods which makes our analysis more rigorous.

Although threshold calibration can affect threshold-dependent metrics like F1-score, its impact may vary for different datasets depending on how much the classes overlap in the probability distributions. However, to the best of our knowledge, the state-of-the-art TL-based CPDP studies use the default classifier threshold to generate the results of threshold-dependent metrics like F1-score, and different CPDP methods might require different types of calibration. Therefore, for fair comparison, we generate all results with the default classifier thresholds (e.g., 0.5 for logistic regression). However, a proper calibration may improve the performance of a specific method.

8. Conclusion

In this paper, we have proposed OCM to extract core matrices of the data in a reduced dimension, which preserves the maximum achievable variance, alleviates multicollinearity, and is low-cost and parameter-free. To perform CPDP, we have extended OCM and proposed a transfer learning method, namely OCMTL, which is able to find the shared components between the source and target that have high discrimination ability. To select the desired number of latent features automatically, we propose a feature selection mechanism that demonstrates further effectiveness of OCMTL. Rigorous experimentation on four benchmark CPDP datasets demonstrates that OCMTL performs better compared to the state-of-the-art methods, considering different metrics. For example, considering AUC across all datasets, the *overall-winner* is OCMTL (25 out 66 pairs) and its closest competitor is TCA+ (10 out 66 pairs); the *average-score* of JCSL (0.693) is the closest to OCMTL (0.711); and DMDA_JFR has the most wins (23 out 66) with OCMTL in terms of *win/tie/loss*. Note that there is no single method that performs equally well in comparison to OCMTL, considering these three criteria (mentioned before), which ensures the superiority of OCMTL. We also perform the Friedman–Nemenyi test and found that there are significant differences among the compared methods, where OCMTL ranks first in most of the cases in terms of AUC. It is observed that the performance in terms of AUC (threshold-independent) is much better than that of F1-score (threshold-dependent), which indicates that, the incorporation of a proper calibration might improve the performance of OCMTL. We aim to address this issue in a separate study in the future.

It is noteworthy to mention that the proposed feature selection method may also be used with other TL methods that produce latent features according to their importance. OCMTL can only be used for homogeneous transfer learning in its current design. We aim to extend our work to design a heterogeneous TL method. The advantageous properties of both OCM and OCMTL that we discussed before make them suitable for many general-purpose ML (such as within-project defect prediction) and TL applications (such as sentiment classification

and object recognition), respectively, in addition to their applicability in SDP. Moreover, we hope OCMTL can be incorporated in applications where explainability is desired, as it considers both the alignment and the variance of the source and target. We plan to address all of these works in future.

The major limitation of OCMTL is that it might not work well when the data is very sparse, as it involves the computation of the covariance of the data. Again, the alignment of the most dominant component(s) of the individual source and target is required for the convergence of OCMTL, though it is shown that OCMTL can capture all shared components if they have discrimination ability. Therefore, we believe the convergence can also be proven in a different manner without the aforementioned assumption. We aim to address these issues in the future.

CRediT authorship contribution statement

Shartaz Sajid Nahid: Writing – original draft, Visualization, Methodology, Formal analysis, Data curation, Conceptualization. **Md. Shariful Islam:** Validation. **Md. Arman Hossain:** Software, Data curation. **Muhammad Mahbub Alam:** Writing – review & editing. **Mohammad Shoyaib:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Conceptualization.

Declaration of competing interest

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

Acknowledgment

This research is supported by the fellowship from ICT Division, Ministry of Posts, Telecommunications and Information Technology, Bangladesh. No - 56.00.0000.000.052.33.0003.24-112; Dated 03.06.2025.

Appendix. Source code for the core part of OCMTL

The implementation of OCMTL is simple, and the MATLAB code for the core part of OCMTL is given below. Here, the variables X_s, X_t, n and k denote the source data, target data, number of features and the size of the latent space.

```
U_Ct = orth(rand(n, k)); U_Cs = U_Ct;
for it=1:100
    U_Ct_old = U_Ct, U_Cs_old = U_Cs;
    [U_As,~] = svds(X_s * U_Ct, k);
    [U_Cs,~] = svds(X_s' * U_As, k);
    [U_At,~] = svds(X_t * U_Cs, k);
    [U_Ct,~] = svds(X_t' * U_At;, k);
    if norm(U_Cs_old-U_Cs,'fro')/norm(
        U_Cs_old,'fro')<0.001 && norm(
        U_Ct_old-U_Ct,'fro')/ norm(U_Ct_old,'
        fro')<0.001
        break;
    end
end
training_data = X_s * U_Ct; testing_data =
    X_t * U_Ct;
```

References

[1] C. Ebert, C. Jones, Embedded software: Facts, figures, and future, Computer 42 (4) (2009) 42–52.
[2] Y. Ma, G. Luo, X. Zeng, A. Chen, Transfer learning for cross-company software defect prediction, Inf. Softw. Technol. 54 (3) (2012) 248–256.
[3] S. Sharmin, M.R. Arefin, M. Abdullah-Al Wadud, N. Nower, M. Shoyaib, SAL: An effective method for software defect prediction, in: ICCIT, IEEE, 2015, pp. 184–189.
[4] F. Wu, X. Jing, Y. Sun, J. Sun, L. Huang, F. Cui, Y. Sun, Cross-project and within-project semisupervised software defect prediction: A unified approach, IEEE Trans. Reliab. 67 (2) (2018) 581–597.
[5] M. Pinzger, N. Nagappan, B. Murphy, Can developer-module networks predict failures? in: SIGSOFT FSE, ACM, 2008, pp. 2–12.
[6] T. Zimmermann, N. Nagappan, H.C. Gall, E. Giger, B. Murphy, Cross-project defect prediction: a large scale experiment on data vs. domain vs. process, in: ESEC/SIGSOFT FSE, ACM, 2009, pp. 91–100.
[7] A. Abdu, Z. Zhai, H.A. Abdo, S. Lee, M.A. Al-masni, Y.H. Gu, R. Algabri, Cross-project software defect prediction based on the reduction and hybridization of software metrics, Alex. Eng. J. 112 (2025) 161–176.
[8] S. Hosseini, B. Turhan, D. Gunarathna, A systematic literature review and meta-analysis on cross project defect prediction, IEEE Trans. Softw. Eng. 45 (2) (2017) 111–147.
[9] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition, Morgan Kaufmann, Elsevier, 2011.
[10] Z. Xu, S. Pang, T. Zhang, X. Luo, J. Liu, Y. Tang, X. Yu, L. Xue, Cross project defect prediction via balanced distribution adaptation based transfer learning, J. Comput. Sci. Tech. 34 (5) (2019) 1039–1062.
[11] J. Nam, S.J. Pan, S. Kim, Transfer defect learning, in: ICSE, IEEE Computer Society, 2013, pp. 382–391.
[12] S. Qiu, L. Lu, S. Jiang, Joint distribution matching model for distribution-adaptation-based cross-project defect prediction, IET Softw. 13 (5) (2019) 393–402.
[13] Q. Zou, L. Lu, Z. Yang, X. Gu, S. Qiu, Joint feature representation learning and progressive distribution matching for cross-project defect prediction, Inf. Softw. Technol. 137 (2021) 106588.
[14] Z. Li, X. Jing, F. Wu, X. Zhu, B. Xu, S. Ying, Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction, Autom. Softw. Eng. 25 (2) (2018) 201–245.
[15] N. Xiao, L. Zhang, X. Xu, T. Guo, H. Ma, Label disentangled analysis for unsupervised visual domain adaptation, Knowl.-Based Syst. 229 (2021) 107309.
[16] S.J. Pan, I.W. Tsang, J.T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE Trans. Neural Netw. 22 (2) (2011) 199–210.
[17] M. Long, J. Wang, G. Ding, J. Sun, P.S. Yu, Transfer feature learning with joint distribution adaptation, in: ICCV, IEEE Computer Society, 2013, pp. 2200–2207.
[18] J. Wang, Y. Chen, S. Hao, W. Feng, Z. Shen, Balanced distribution adaptation for transfer learning, in: ICDM, IEEE Computer Society, 2017, pp. 1129–1134.
[19] B. Sun, J. Feng, K. Saenko, Correlation alignment for unsupervised domain adaptation, in: Domain Adaptation in Computer Vision Applications, in: Advances in Computer Vision and Pattern Recognition, Springer, 2017, pp. 153–171.
[20] B. Fernando, T. Tommasi, T. Tuytelaars, Joint cross-domain classification and subspace learning for unsupervised adaptation, Pattern Recognit. Lett. 65 (2015) 60–66.
[21] D.A. K., R.K. Sanodiya, B.R. Jose, J. Mathew, Visual domain adaptation through locality information, Eng. Appl. Artif. Intell. 123 (Part A) (2023) 106172.
[22] Y. Lu, W.K. Wong, C. Yuan, Z. Lai, X. Li, Low-rank correlation learning for unsupervised domain adaptation, IEEE Trans. Multimed. (2023).
[23] S. Akhter, M.M. Alam, M.S. Islam, M.A. Momen, M.S. Islam, M. Shoyaib, Low-cost orthogonal basis-core extraction for classification and reconstruction using tensor ring, Pattern Recognit. 154 (2024) 110548.
[24] J. Li, F. Huang, Guaranteed simultaneous asymmetric tensor decomposition via orthogonalized alternating least squares, 2018, arXiv preprint arXiv:1805.10348.
[25] B. Turhan, T. Menzies, A.B. Bener, J.S.D. Stefano, On the relative value of cross-company and within-company data for defect prediction, Empir. Softw. Eng. 14 (5) (2009) 540–578.
[26] L. Yu, A. Mishra, Experience in predicting fault-prone software modules using complexity metrics, Qual. Technol. Quant. Manag. 9 (4) (2012) 421–434.
[27] P. He, B. Li, X. Liu, J. Chen, Y. Ma, An empirical study on software defect prediction with a simplified metric set, Inf. Softw. Technol. 59 (2015) 170–190.
[28] C. Liu, D. Yang, X. Xia, M. Yan, X. Zhang, A two-phase transfer learning model for cross-project defect prediction, Inf. Softw. Technol. 107 (2019) 125–136.
[29] M.A. Hossain, S. Akhter, M.S. Islam, M.M. Alam, M. Shoyaib, Can a simple approach perform better for cross-project defect prediction? in: ENASE, SCITEPRESS, 2024, pp. 328–335.
[30] J. Niu, Z. Li, C. Qi, Correlation metric selection based correlation alignment for cross-project defect prediction, in: IUCC/CIT/DSCI/SmartCNS, 2021, pp. 490–495.
[31] M. Nevendra, P. Singh, Meta network attention-based feature matching for heterogeneous defect prediction, Autom. Softw. Eng. 32 (1) (2025) 1–32.

[32] J. Niu, Z. Li, H. Chen, X. Dong, X. Jing, Data sampling and kernel manifold discriminant alignment for mixed-project heterogeneous defect prediction, Softw. Qual. J. 30 (4) (2022) 917–951.
[33] P. Zhu, A.V. Knyazev, Angles between subspaces and their tangents, J. Num. Math. 21 (4) (2013) 325–340.
[34] T. Menzies, J. Greenwald, A. Frank, Data mining static code attributes to learn defect predictors, IEEE Trans. Softw. Eng. 33 (1) (2007) 2–13.
[35] K.-J. Bathe, J.T. Oden, E.L. Wilson, Formulations and Computational Algorithms in Finite Element Analysis: US-Germany Symposium, MIT Press (MA), 1977.
[36] B. Fernando, A. Habrard, M. Sebban, T. Tuytelaars, Unsupervised visual domain adaptation using subspace alignment, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 2960–2967.
[37] M. D'Ambros, M. Lanza, R. Robbes, Evaluating defect prediction approaches: a benchmark and an extensive comparison, Empir. Softw. Eng. 17 (4–5) (2012) 531–577.
[38] R. Wu, H. Zhang, S. Kim, S. Cheung, Relink: recovering links between bugs and changes, in: SIGSOFT FSE, ACM, 2011, pp. 15–25.
[39] T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, B. Turhan, The promise repository of empirical software engineering data, 2012.
[40] M.J. Shepperd, Q. Song, Z. Sun, C. Mair, Data quality: Some comments on the NASA software defect datasets, IEEE Trans. Softw. Eng. 39 (9) (2013) 1208–1215.
[41] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: A library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874.
[42] C.X. Ling, J. Huang, H. Zhang, AUC: a statistically consistent and more discriminating measure than accuracy, in: IJCAI, Morgan Kaufmann, 2003, pp. 519–526.
[43] D. Chen, W. Fu, R. Krishna, T. Menzies, Applications of psychological science for actionable analytics, in: ESEC/SIGSOFT FSE, ACM, 2018, pp. 456–467.
[44] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.