Adversarial Learning for Cross-Project Semi-Supervised Defect Prediction

YING SUN1, XIAO-YUAN JING1,2,3, FEI WU2, JUANJUAN LI2, DANLEI XING1, HAOWEN CHEN3, AND YANFEI SUN2

1 School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
2 College of Automation, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
3 School of Computer Science, Wuhan University, Wuhan 430072, China

Corresponding authors: Xiao-Yuan Jing (jingxy_2000@126.com) and Fei Wu (wufei_8888@126.com)

This work was supported in part by the NSFC-Key Project of General Technology Fundamental Research United Fund under Grant U1736211, in part by the Natural Science Foundation Key Project for Innovation Group of Hubei Province under Grant 2018CFA024, in part by the National Natural Science Foundation of China under Grant 61702280, in part by the National Postdoctoral Program for Innovative Talents under Grant BX2018046, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20170900, in part by the China Postdoctoral Science Foundation under Grant 2019M661901, in part by the Jiangsu Planned Projects for Postdoctoral Research Funds under Grant 2019K024, in part by the CCF-Tencent Open Fund WeBank Special Funding under Grant CCF-WeBankRAGR20190104, and in part by the Scientific Research Starting Foundation for Introduced Talents in the Nanjing University of Posts and Telecommunications (NUPTSF), under Grant NY217009.

ABSTRACT Cross-project defect prediction (CPDP) aims to build a prediction model on existing source projects and predict the labels of target project. The data distribution difference between different projects makes CPDP very challenging. Besides, most existing CPDP methods usually require sufficient and labeled data. However, acquiring lots of labeled data for a new project is difficult while obtaining the unlabeled data is relatively easy. A desirable approach is building a prediction model on unlabeled data and labeled data. CPDP in this scenario is called cross-project semi-supervised defect prediction (CSDP). Recently, generative adversarial networks have achieved impressive results with these strong ability of learning data distribution and discriminative representation. For effectively learning the discriminative features of data from different projects, we propose a Discriminative Adversarial Feature Learning (DAFL) approach for CSDP. DAFL consists of feature transformer and project discriminator, which compete with each other. A feature transformer tries to generate feature representation, which learns the discriminative information and preserves intrinsic structure inferred from both labeled and unlabeled data. A project discriminator tries to discriminate source and target instances on the generated representation. Experiments on 16 projects show that DAFL performs significantly better than baselines.

INDEX TERMS Cross-project defect prediction, adversarial learning, semi-supervised learning.

I. INTRODUCTION Software defect prediction (SDP) [1]–[8] is an important software quality assurance step of predicting the defect-proneness in software project development history. Many prior SDP studies predict the fault of a new instance within the same project, which is called within-project defect prediction (WPDP) [9]–[13]. In recent studies, machine learning models are successfully applied to identify defective instances, e.g., canonical correlation analysis [14], [15], neural networks [16], [17], and deep learning (DL) [18]–[22]. Prior studies have shown that a useful machine learning model needs to be trained by using sufficient and complete data. However, it is a challenging problem that a new project with limited historical data could perform the prediction model well. When we do not have sufficient amount of historical data, cross-project defect prediction (CPDP) [23] is a satisfactory solution, which refers to building the prediction model trained by the data from source projects and then predicting the label of a target project.

However, in fact, achieving a satisfactory CPDP often is challenging. Zimmermann et al. [24] evaluated CPDP performance on 12 projects (622 cross-project pairs), and only 21 pairs performed well. The data from different projects...
usually exhibit significant distribution gap, which violates a basic requirement of most machine learning modeling technology, i.e., the similar distribution assumption [25], [26]. Many studies aim to decrease the feature distribution difference from different projects in CPDP [25], [26]. These researches has been focused on feature (metric) selection. Subspace learning is a popular approach. With subspace learning, the distribution gap can be reduced and the transformed features can acquire the maximum correlations by learning corresponding transforms. Deep neural network (DNN) [27]–[30] models have provided scalable nonlinear transformations for feature representations in image classification scenario, and DNN has received extensive development. Recently, generative adversarial networks (GANs) [31] have been proposed, which build an adversarial modeling framework by simultaneously training two models. To learn good feature representation of the input data, GANs perform a minimax game by making two models compete with each other. GANs have excellent ability of data distribution generation and feature representation learning. Thus GANs can be used for modeling the data distribution of different projects. Then we can learn the common discrimination representation of inter-class instances and boost the correlation learning of intra-class instances in different projects.

Most existing CPDP approaches focus on building prediction model by using only labeled data. In software development practice, the availability of labeled data is limited due to reasons such as high cost, lack of budget, and time limitation [32]–[34]. It is difficult for a new project to obtain sufficient and labeled data. Therefore, in such cases, we can not use the regular supervised techniques for CPDP. Fortunately, it is relatively easy for obtaining unlabeled data. And then with the help of the sufficient unlabeled data from external project, CPDP can be performed better.

In order to address the challenges of distribution difference between different projects and limited number of labeled data, we propose a new approach, termed Discriminative Adversarial Feature Learning (DAFL) for CSDP. Fig. 1 illustrates the general framework of DAFL. DAFL consists of three major parts: feature mapping, feature transformer, and project discriminator. We adopt four-layer feed-forward neural network (FNN) [35] as the feature mapping in DAFL to nonlinearily map the respective features from different projects to a common subspace. Feature transformer generates inter-class separated and intra-class correlated representation instances (i.e., the instances from the same classes are close and the instances form different classes are separate) based on labeled data and lots of unlabeled data from different projects in the common subspace. Feature transformer performs the representation task by feature discrimination and feature structure preservation. The feature structure preservation is accomplished by triplet constraint. The maximum correlation of transformed features between source and target projects is vital for CPDP. It is also significant that whether the features are statistically indistinguishable. To achieve this goal, we apply project discriminator to identify the source project and target project of a transformed instance. Project discriminator tries to distinguish the project label (i.e., the label with which we can distinguish the instances from source project or target project) and in this way guides the learning of the project transformer. Feature transformer and project discriminator are trained under the adversarial learning framework. Therefore, we can ensure that the transformed instances of different projects are both discriminative across different classes within a project and intrinsic structure invariant across projects.

The main contributions of this paper are as follows:
1) We introduce the adversarial learning framework into cross-project defect prediction to better address the data distribution difference problem of different projects, which can effectively maximize the feature correlation over the different projects. The discriminator judges the project label from source project or target project. The transformer learns the intra-class correlation and inter-class discrimination from both the within-project and cross-projects. Discriminator and transformer
II. RELATED WORK

A. CROSS-PROJECT DEFECT PREDICTION (CPDP)

To solve the shortage of historical data for target project, the researchers turn their attentions on external projects and propose CPDP. Zimmermann et al. [24] first attempted to conduct CPDP model. They ran 622 cross-project predictions and only 21 had satisfactory experiment results. The success rate is only 3.4%. Thus they concluded that CPDP was still a difficult challenge. He et al. [36] concluded that the selection of training data is a vital step for the constructing of CPDP model. They selected suitable training data for target project, and supported the conclusion that the prediction results of CPDP was related with the distributional characteristics of datasets. Turhan et al. [37] proposed a filter for companies that lack sufficient historical data to build defect prediction model. They analyzed the applicability of building prediction model by cross-company data. And then they selected ten nearest neighbors instances for each unlabeled test instance by k-nearest neighbor algorithm. The experiment demonstrates that this method is effective for building predictors by small sample data. Peters et al. [38] extended LACE2 which reduced the amount of data shared by using multi-party data sharing. Also, the obfuscation algorithm is applied to hiding project details to realize privacy-preserving for CPDP. Xia et al. [39] proposed a hybrid model reconstruction approach (HYDRA), which could iteratively learn new classifiers and compositions of classifiers to collectively better capture generalizable properties. By those iterative learning of various classifiers, HYDRA could alleviate the challenge of distribution difference between different projects. Poon et al. [40] used a credibility theory based Naïve Bayes classifier. It provides a credibility factor which can establish a novel reweighing mechanism so that the source project data can adapt to the target project data whilst retaining their own data pattern. Herbold et al. [41] reproduced 24 methods and evaluated the performance on CPDP. They determined that

2) DAFL is a semi-supervised approach that performs triplet sampling from labeled and unlabeled data. The triplet constraints are enforced on the feature transformer for making full use of intrinsic geometrical structure information from unlabeled data and class label information from labeled data.

3) We perform the proposed approach on NASA, AEEEM, and PROMISE datasets. The experiment results show that our approach achieves competitive performance compared to the state-of-the-art CPDP and semi-supervised defect prediction methods.

The rest of paper is organized as follows: Section II outlines related works on CPDP and semi-supervised defect prediction. In Section III, we describe our proposed method DAFL. Experimental settings and results are demonstrated in Sections IV and V. We outline different threats to validity in Section VI. Section VII concludes this paper.

B. SEMI-SUPERVISED DEFECT PREDICTION (SSDP)

To solve the shortage of labeled data, a feasible solution is making use of the information of both labeled and unlabeled software instances, and SSDP methods have been presented [33], [44]–[46]. Seliya and Khoshgoftaar [47] proposed an expectation maximization based semi-supervised method. It estimates the model parameters and classification probability of unlabeled data. Lu et al. [48], [49] proposed a method called fitting-the-confident-fits with multidimensional scaling (FTcF-MDS). Firstly, FTcF first labels the unlabeled data, and then a supervised learner is trained by using all the labeled data. Multidimensional scaling is used to reduce the feature dimension before FTcF is initiated. Catal [50] made an assessment about four semi-supervised classification methods, including low-density separation (LDS) and so on. Experimental results showed that LDS-based prediction approach is useful for SDP when the fault data are limited. Ma et al. [51] proposed random undersampling tri-training (RusTri) method, which employs the sampling strategy to resample the original training set and tri-training method to update training set. Thung et al. [52] proposed a SSDP method which labels a small data subset and then used labeled and unlabeled data in the learning process. He et al. [53] proposed a method named extRF. ExtRF extends the random forest algorithm by self-training paradigm and also employs change information to improve prediction performance. Zhang et al. [33] proposed a nonnegative sparse graph based label propagation (NSGLP) method for defect prediction and classification. NSGLP solves class-imbalanced problem of training data by Laplacian score sampling strategy and sparse representation. The next step is to learn a relationship graph by the using of a nonnegative sparse algorithm. Based on the relationship graph, NSGLP employs a label propagation algorithm to predict defect-proneness of unlabeled instances. Yu et al. [46] proposed a semi-supervised clustering-based data filtering (MsTrA+) method to filter out irrelevant cross-company data.

Recently, Wu et al. [32] for the first time studied the cross-project semi-supervised defect prediction (CSDP) problem and proposed the cost-sensitive kernelized semi-supervised dictionary learning (CKSDL) method. They introduced

CamargoCruz09 [42] perform best. Jing et al. [43] proposed a method named SSTCA + ISDA for both WPDP and CPDP. They introduced subclass discriminant analysis into CPDP and WPDP to solve the class imbalance problem. Meanwhile, the semi-supervised transfer component analysis method is employed to make the distributions of source and target data similar. Zhou et al. [8] investigated existing CPDP models between 2002 and 2017 years. They concluded that the prediction performance of simple module size models are superior to most of the existing CPDP models in the literature. There is an assumption in most existing CPDP methods that the available source project data are well labeled. In fact, labeling data requires expert judgment and labeling lots of data is time-consuming.
semi-supervised dictionary learning into SDP field. CKSDL uses kernel mapping to enhance the separability of data and cost-sensitive technique to ease misclassification problem. Different from existing CSDP methods, DAFL utilizes the adversarial learning, which can learn the discriminative feature of unlabeled and labeled data and has stronger feature learning ability, to improve prediction ability for defect prediction model.

C. ADVERSARIAL LEARNING

Goodfellow et al. [31] proposed adversarial learning in GANs. The original framework consists of two major components: a generative model $G$ and a discriminative model $D$, which have two opposite training goals. To learn the distribution model. The generator $G$ is a generative model to generate instances that $D$ cannot distinguish those instances and the real instances during the generation process; $D$ is a discriminative model to correctly distinguish the real instances and fake instances during the discrimination process. $D$ and $G$ play a minimax game as follows

$$
\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x \sim p_{data}(x)} \left[ \log D(x) \right] \\
+ \mathbb{E}_{z \sim p_{z}(z)} \left[ \log (1 - D(G(z))) \right]
$$

(1)

where $x$ denotes the true data, $z$ denotes the noise data. The generator $G$ defines a probability distribution $p_{g}$ as the distribution of the instances $G(z)$ obtained when $z \sim p_{z}$. Goodfellow et al. [31] proved that this game has a global optimum for $p_{g} = p_{data}$. Therefore, the function $V(D,G)$ could converge to a good estimator of $p_{data}$.

Since the adversarial learning was first introduced, it has gradually been a hot topic and used in many fields, including image generation [54], [55], cross-modal retrieval [56]–[58], etc. Wang et al. [58] successfully applied adversarial learning into cross-modal retrieval for different modalities. They first used neural networks for feature transforming. Then they sought an effective common subspace which can regularize the data distributions based on adversarial learning. Adversarial learning consists of two adversarial processes: a feature transformer and a modality classifier. The experiment shows that their method narrows down the distribution gap between different modalities either maximizes correlations of similar instances. Inspired by this work, adversarial learning is introduced into CPDP. We first use adjustable neural networks for feature dimensionality reduction. Then we use adversarial learning framework for narrowing the data distribution gap from different projects. Furthermore, inspired by triplet ranking loss [59], which is effective for feature learning, we make an improvement on general adversarial learning.

III. OUR APPROACH

A. PROBLEM FORMULATION

Assume that $S = \{s_1, s_2, \ldots, s_m\} \in \mathbb{R}^{d \times m}$ and $T = \{t_1, t_2, \ldots, t_n, t_{n+1}, \ldots, t_{n+n_u}\} \in \mathbb{R}^{d \times (n+n_u)}$ separately denote the source project data and target project data, where $m$ and $n + n_u$ denote the numbers of the software instances in $S$ and $T$, $n_u$ denotes the number of unlabeled instances from target project, $d$ denotes the number of metrics. We define $T_u = \{t_{n+1}, \ldots, t_{n+n_u}\}$ as the unlabeled dataset from target project. The source data are all unlabeled. We also define label set of labeled data from target project as $Y^t = \{y^t_1, y^t_2, \ldots, y^t_n\} \in \mathbb{R}^{c \times n}$, $c$ is the number of classes.

B. DATA PREPROCESSING

1) DATA NORMALIZATION

The value of various software metrics usually have a wide range, so we conduct data normalization on software metrics. We normalize the data by using the commonly used min-max normalization [60]. And then all the values are transformed to the interval $[0, 1]$. Given a metric $a_i$ of an instance, the normalized value $\tilde{a}_i$ is computed as

$$
\tilde{a}_i = \frac{a_i - \min(a)}{\max(a) - \min(a)}
$$

(2)

where $\max(a)$ and $\min(a)$ are maximum and minimum values of $a$.

2) PSEUDO LABEL FOR SOURCE PROJECT

In order to alleviate the influence of class-imbalance problem and learn cost-sensitive discriminative information from unlabeled and labeled data, we use simplified cost-sensitive local collaborative representation (CLCR) [61] to infer the pseudo label information for unlabeled data. We simplify the neighboring selecting process of CLCR and then refer to the remaining process of CLCR. The collaborative representation has been introduced into SDP and proved the effectiveness for SDP. Inspired by the discriminant ability of collaborative representation, CLCR also involves the costsensitive factor in the representation coefficients to infer the label information.

For a given $i$-th unlabeled instance $s_i$ from $S$, we obtain the representation coefficients as follows

$$
\min_{\alpha} \left\| s_i - T \bar{C}^{-1} \alpha \right\|_2 + \mu \| \alpha \|_2
$$

(3)

where $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$ is the collaborative representation coefficient vector, $T$ denotes vector or matrix transposition in our paper, and $\mu$ is a regularization coefficient. $\bar{C}^{-1} = \text{diag}(\text{cost}_1, \text{cost}_2, \ldots, \text{cost}_n)^{-1}$ is cost matrix and its element can be defined as follows

$$
\text{cost}_i = \begin{cases} 
\text{num(defective)}, & \text{if } t_i \text{ is defective} \\
\text{num(defect-free)}, & \text{if } t_i \text{ is defect-free} \\
1, & \text{otherwise}
\end{cases}
$$

(4)

where $\text{num(defective)}$ and $\text{num(defect-free)}$ are the numbers of defective instances and defect-free instances. The sparse coefficient vector $\alpha$ can be computed by the following formula:

$$
\alpha = ((T \bar{C}^{-1})^T T \bar{C}^{-1} + \mu I)(T \bar{C}^{-1})^T s_i
$$

(5)

Considering that $\| s_i - t_i \alpha \|_2$ is the class specific representation residual, and $\| \alpha \|_2$ is also associated with class
information, they can bring discrimination information for classification, the regularized residual of each class is calculated as
\[ e_i = \| s_i - t_i \alpha_i \|_2 / \| \alpha_i \|_2 \]  
(6)

Then the instance \( s_i \) is assigned to the \( j \)-th class
\[ \hat{y}_j^i = \arg\min\{e_i\} \]  
(7)

After CLCR, the pseudo label set of source project we obtained is defined as \( \hat{Y}^S = \{ \hat{y}_1^s, \hat{y}_2^s, \ldots, \hat{y}_n^s \} \subseteq R^{c \times m} \). And we also adopt a standard random oversampling [62] to alleviate the influence of class-imbalance problem.

C. ADVERSARIAL LEARNING

The framework of DAFL (Discriminative Adversarial Feature Learning) is shown in Fig. 1. In feature mapping, the datasets \( S \) and \( T \) have been transformed in common subspace \( F \). Although the data between source project and target project have significant difference, the abundant parameters of fully-connected layers can ensure the excellent ability of representations. Then, DAFL plays the minimax game between the feature transformer and project discriminator, and these two processes steer the feature representation learning. Specifically, we decompose the learning processes into three parts: project discriminator is utilized to minimize the distribution gap of representations between different projects; feature discrimination is used to ensure learned feature representations from different classes are discriminative; feature structure preservation, which maximizes the distances between inter-class instances and minimizes the distances among intra-class instances in different projects simultaneously.

1) FEATURE MAPPING

CPDP aims to extract common knowledge from source project and transfer the knowledge to target project. As different projects have different data distributions, if we training a prediction model using data from external projects (source project) directly, CPDP cannot generalize well to the target project [24]. We aim to find a common subspace for minimizing the distance between source and target projects while preserving the original data properties. The source project data \( S \) and target data \( T \) can be mapped to \( F_S = f_S (S; \theta_S) \) and \( F_T = f_T (T; \theta_T) \). \( f_S (S; \theta_S) \) and \( f_T (T; \theta_T) \) are mapping functions, \( F_S \subseteq R^{c \times m} \) and \( F_T \subseteq R^{d \times (n + m)} \) are the transformed instances in subspace \( F \), \( \theta_S \) and \( \theta_T \) are adjustable parameters. Inspired by the subspace learning methods, \( F_S \) and \( F_T \) in our method are realized as feed-forward neural networks. FNN has enough capacity of feature representations considering the large difference of complex distribution between different projects. We adopt four-layer FNN as the feature mapping in DAFL to nonlinearly transform the features from different projects into a common subspace.

2) PROJECT DISCRIMINATOR

We first define a discriminator \( D \) with parameters \( \theta_D \).

As an anchor. The target instance \( t_i \) is selected as an anchor. The target instance \( t_i^+ \) is assigned as a positive matching with the closest distance compared with other target
instances, and \( t_j^- \) with the farthest distance compared with other target instances is assigned as a negative mismatching. Similarly, given a target instance \( t_i \), we have \( (t_i, s_i^+) \) and \( (t_i, s_i^-) \). We also define the constraints for the target instances as \( (t_i, s_i^+), (t_i, s_i^-) \) by utilizing the label information from target labeled data. \( s_i^+ \) is the instance farthest from \( t_i \) in same class, and \( s_i^- \) is the instance closest to \( t_i \) from different class.

Finally, we build the sets of triplets \( \{s_i, t_i^+, t_i^-\} \) for per source instance, \( \{t_i, s_i^+, s_i^-\} \) and \( \{t_i, t_i^+, t_i^-\} \) for per target instance. The \( L_2 \) norm is used to compute the distance between all mapped instances after feature mapping:

\[
l_2(s, t) = \|f_S(s; \theta_S) - f_T(t; \theta_T)\|_2
\]

Thus, we compute the loss between different projects using the following equations:

\[
L_{str} S (\theta_S) = \sum_{i,j,k} l_2\left(s_i, t_j^+\right) + \lambda \cdot \max\left(0, \gamma - l_2\left(s_i, t_k^-\right)\right)
\]

\[
L_{str} T (\theta_T) = \sum_{i,j,k} \left\{l_2\left(t_i, s_j^+\right) + \lambda \cdot \max\left(0, \gamma - l_2\left(t_i, s_k^-\right)\right)\right\}
\]

(11)

(12)

where the hyper-parameter \( \gamma \) is a distance threshold value, \( \lambda \) is a balance factor.

Then the overall feature structure preservation loss across different projects is defined by the following equation:

\[
L_{str} (\theta_S, \theta_T) = L_{str} S (\theta_S) + L_{str} T (\theta_T)
\]

(13)

Based on the above equations, the loss function of the feature generation is formulated as the discrimination loss \( L_D \) by using inter-class discrimination information and intra-class correlation information between different projects.

\[
L_D (\theta_S, \theta_T, \theta_Y) = L_{fd} + \beta L_{str}
\]

(14)

where the hyper-parameter \( \beta \) controls the contributions of the \( L_{fd} \) and \( L_{str} \).

4) OPTIMIZATION
The optimization goals of our functions are adversarial, which are trained as a min-max game of the following processes:

\[
\hat{\theta_S}, \hat{\theta_T}, \hat{\theta_Y} = \arg\min_{\theta_S, \theta_T, \theta_Y} \left( L_G (\theta_S, \theta_T, \theta_Y) - L_D (\hat{\theta_D}) \right)
\]

(15)

\[
\hat{\theta_D} = \arg\max_{\theta_D} \left( L_G (\hat{\theta_S}, \hat{\theta_T}, \hat{\theta_Y}) - L_D (\theta_D) \right)
\]

(16)

IV. EXPERIMENT
A. RESEARCH QUESTIONS
We investigate two questions in this paper as follows:

**RQ1:** Is the prediction performance improved when we use DAFL?

**RQ2:** Whether each component of DAFL is effective or not?

For RQ1, we compare DAFL with six state-of-the-art SSDP methods and four CPDP methods. The SSDP methods include FTcF-MDS [48], LDS [50], RusTr [51], NSGLP [33], MsTrA+ [46], and CKSDL [32]. MsTrA+ and CKSDL are two SSDP methods. Four typical CPDP methods include ManualDown [8], CamargoCruz09 [42], NN-filter [37], and SSTCA + ISDA [43].

For RQ 2, we investigate whether the important components of DAFL are effective or not. First, to explore the effect of adversarial learning in DAFL, we display the values of discrimination and adversarial loss form epoch 1 to 10000. Then we compare DAFL and its two variants (DAFL with \( L_{fs} \) only and DAFL with \( L_{str} \) only). Finally, we compare DAFL against the baseline that is DAFL without pseudo labels.

B. DATASETS
We conduct experiments on three widely used software defect prediction datasets: AEEEM [64], NASA [65], PROMISE [66], [67]. Table 1 lists the details about the datasets. A brief introductions on each dataset are mentioned as follows.

The AEEEM dataset consists of Equinox Framework (EQ), Eclipse JDT Core (JDT), Apache Lucene (LC), Mylyn (ML), and Eclipse PDE UI (PDE) projects. It has 61 metrics, including Chidamber&Kemerer (CK) metrics, object-oriented metrics, churn of source code metrics, and entropy of source code metrics.
TABLE 1. Details of project used in experiments.

| Dataset | Project | Number of instances | Percentage of defective instances(%) | Number of metrics |
|---------|---------|---------------------|---------------------------------------|-------------------|
| NASA    | EQ      | 324                 | 39.81                                 | 61                |
|         | JDT     | 997                 | 20.66                                 | 61                |
|         | LC      | 691                 | 9.26                                  | 61                |
|         | ML      | 1862                | 13.16                                 | 61                |
|         | PDE     | 1497                | 13.96                                 | 61                |
| AEEEM   | CM1     | 505                 | 9.50                                  | 37                |
|         | MW1     | 403                 | 7.69                                  | 37                |
|         | PC1     | 1107                | 6.97                                  | 37                |
|         | PC3     | 1563                | 10.24                                 | 37                |
|         | PC4     | 1458                | 12.21                                 | 37                |
| PROMISE | Ant v1.7| 745                 | 22.30                                 | 20                |
|         | Camel v1.6| 965              | 19.50                                 | 20                |
|         | Poi v3.0| 442                 | 63.60                                 | 20                |
|         | Tomcat v6.0| 858             | 9.00                                  | 20                |
|         | Xalan v2.6| 885              | 46.40                                 | 20                |
|         | Xerces v1.3| 453             | 15.23                                 | 20                |

The NASA dataset was collected by the NASA metrics data program. Each project of NASA represents a subsystem from NASA. We select five projects from NASA dataset, and these five projects have 37 common metrics, which include code size and complexity etc. [68].

The PROMISE dataset was prepared by Jureczko et al. [66]. In the PROMISE dataset, not all the projects have the same set of metrics. Hence, we select six open-source projects that have 20 common metrics. The metrics contain McCabe’s cyclomatic metrics (e.g., Average Cyclomatic Complexity), CK metrics (e.g., Coupling between Object Classes) and other object-oriented metrics (e.g., Depth of Inheritance Tree).

C. EVALUATION MEASURE

In this paper, we use two measures to evaluate the performance of DAFL: F-measure and G-measure. F-measure and G-measure are well-known measures [8], [15], [38], [46], [69], [70], which are applied to evaluate the prediction performance. We follow previous studies and use F-measure and G-measure as the indicators in our study. The values of measures range from 0 to 1. The higher the value means, the better the performance is.

F-measure is the harmonic mean of recall and precision and balances out the recall-precision tradeoff. Recall (Pd) is a statistical measure defined as \( TP / (TP + FN) \), where \( TP \) and \( FN \) mean True Positive and False Negative. Precision (Pre) is a statistical measure defined as \( TP / (TP + FP) \), \( FP \) means False Positive. F-measure is defined as \( \frac{1+\alpha^2}{\alpha^2} \times \frac{Pd \times Pre}{Pd + \alpha^2 \times Pre} \), where \( \alpha \) is a parameter to decide the relative importance of the Precision over the recall. We usually set the parameter \( \alpha \) as 1 [15]. G-Measure takes both recall and specificity into consideration and is the geometric mean of recall and specificity. Specificity is a statistical measure defined as \( TN / (TN + FP) \), where \( TN \) means True Negative. G-measure is defined as \( \frac{TP \times specificity}{TP \times specificity + FN} \). \( TP, FN, FP \) and \( TN \) are defined in Table 2.

D. BASELINES FOR COMPARISON

We compare our approach DAFL with ten state-of-the-art methods including six SSDP methods (FTcF-MDS [48], LDS [50], RusTri [51], NSGLP [33], MsTrA+ [46], CKSDL [32]) and four CPDP methods (ManualDown [8], CamargoCruz09 [42], NN-filter [37], and SSTCA + ISDA [43]). Specifically, CKSDL is the first method that apply to CSDP scenario. MsTrA+ is a cross-company semi-supervised method. We consider MsTrA+ as an CSDP method. ManualDown is an unsupervised method, which is a baseline model suggested in [8] for comparison when developing new CPDP model. We consider ManualDown as an unsupervised CPDP baseline. CamargoCruz09 is a classical method which is always ranked among the best statistically significant test results in [41] for all metrics and data sets in CPDP. NN-filter and SSTCA + ISDA are effective CPDP methods. To make fair evaluation with these methods, we re-evaluate the method implementations which provided by authors.

E. EXPERIMENT SETTINGS

We organize the cross-project semi-supervised setting on the same dataset and conduct CSDP. For example, for the AEEEM dataset, when EQ is regarded as the target project, other four projects in AEEEM are regarded as the source project in turn, and four cross-project pairs will be formed. We regard all the instances of the source data as unlabeled data. In the target project, we randomly select a certain percentage (e.g., 20%) instances as labeled data. For tackling the randomness of instance selection, we randomly perform the experiments for 5 times. Finally we report the mean F-measure and G-measure results for each target project.

F. IMPLEMENTATION DETAILS

We employ four-layer feed-forward neural networks to non-linear project source project and target project into a common subspace. We conduct the experiments with different

TABLE 2. Confusion matrix.

|          | Predict as defective | Actual defective | Actual defect-free |
|----------|----------------------|------------------|-------------------|
|          | TP                   | FN                | FP                |

TABLE 3. The number of nodes in each layer.

| Layer | AEEEM | NASA | PROMISE |
|-------|-------|------|---------|
| L1    | 60    | 30   | 20      |
| L2    | 40    | 20   | 15      |
| L3    | 20    | 10   | 10      |
| L4    | 10    | 10   | 10      |
| L5    | 5     | 5    | 5       |
| L6    | 2     | 2    | 2       |
| L7    | 5     | 5    | 5       |
| L8    | 2     | 2    | 2       |
| L9    | 2     | 2    | 2       |
parameters of the four-layer network on AEEEM, NASA, and PROMISE datasets. For the project discriminator, we use the three fully connected layers. Furthermore, we use softmax activation for the discriminator. The number of nodes in each layer in network architecture is summarized in Table 3.

In experiment, we set the batch size as 64 on all the datasets. The model parameters $\mu$, $\gamma$, $\lambda$ and $\beta$ are set as $0.05, 0.1, 0.01$ and $0.1$, separately. The values of $\mu$, $\gamma$ and $\lambda$ are mainly follow the paper of [61] and [58]. The parameter $\beta$ is important that controls the contributions of feature discrimination and feature structure preservation. We search values between 0 and 100 and set the range of values for $\beta$ as $[0.01, 0.1, 1, 10, 100]$. Finally we choose one value that yields the best F-measure and G-measure values at the same time.

V. EXPERIMENT RESULTS

A. ANALYSIS RESULTS OF RQ1

In this section, we compare DAFL with ten methods. Tables 4 and 5 report the F-measure and G-measure values of all baselines on three companies. The mean F-measure and G-measure values corresponding to each target project are reported. We also report the mean values of different target projects in a company in the table. The best values for each target project are highlighted in bold font. From these tables, DAFL obtains significant prediction performance improvement in most of the cases. On average, DAFL obtains the Feasure and G-measure as 0.7034 and 0.6944 and achieves the best average F-measure and G-measure. Our DAFL can significantly outperform the traditional SSDP methods (FtCf-MDS, LDS, RusTri, NSGLP, MsTrA+ and CKSDL) and the state-of-the-art CPDP methods (CamargoCruz09, ManualDown, NN-filter and SSTCA+ ISDA), and this shows the advantage of adversarial learning for CPDP.

Compared with SSDP methods: in most cases, F-measure and G-measure values of DAFL are higher than related methods. The performance improvement compared with competing methods is significant. For example, DAFL improves the mean F-measure by 77.17%, 82.98%, 59.57%, 70.19%, 68.89% and 44.49% over FtCf-MDS, LDS, NSGLP, RusTri, MsTrA+ and CKSDL, respectively. DAFL uses large unlabeled instances from other project, which contains useful information. DAFL can learn the information effectively. And our approach can model the distribution between data form different projects and then effectively reduce the gap between source and target project data. The adversarial techniques have been theoretically and empirically proven to be able to minimize the distribution difference. Thus, DAFL performs better than the baseline methods.

VOLUME 8, 2020

32681

IEEE Access
better than other competing methods. Specifically, compared with two CSDP methods (MsTrA+ and CKSDL), the prediction performance of DAFL is improved. Our approach shows better ability of obtaining effective transformed features. On the one hand, we transform the data from different projects into a common subspace by feature mapping. This process can ensure the data between different projects is correlated. On the other hand, the strong ability of feature learning of DAFL makes the prediction performance of our model improving a lot. Compared with MsTrA+ and CKSDL, our model has stronger feature learning and discrimination ability.

Compared with CPDP methods: ManualDown, CanargoCruz09, NN-filter and SSTCA + ISDA are four baselines which have shown a good performance in recent CPDP surveys [8], [41]. In most cases, our DAFL has better performance than these four baselines. DAFL fully extracts the useful information from both the large amount of unlabeled instances and labeled instances in the common subspace. CanargoCruz09 can only perform data transformation, ManualDown can only utilize the module size feature, NN-filter aims to collect the required local data easily and quickly and is fail to construct a better model, and SSTCA + ISDA cannot use the unlabeled data effectively, which leads to the undesirable performances as compared with DAFL.

1) STATISTICAL SIGNIFICANCE TEST AND EFFECT SIZE TEST

Statistical Significance Test: To statistically analyze the detailed results corresponding to Tables 4 and 5 (5 random running), we perform the non-parametric Friedman test [71] with the Nemenyi test (at a confidence of 95%) to compare the significant difference of the model performance. The evaluation has been used for performance comparison in many studies [15], [41], [72]. For each measure, we compare multiple model over the datasets.

We report the visual representation results of DAFL against competing methods in Fig. 2.(a)-(b). The lower ranking indicates the better performance. In terms of F-measure value, we observe the non-parametric Friedman test value in Fig. 2, DAFL outperforms other competing methods with statistical significance. In the same way, DAFL has the similar conclusion on F-measure. The results show that DAFL achieves better prediction performance than baselines.

| Table 6. | Mappings of Cliff’s delta values to effectiveness levels. |
| --- | --- |
| Cliff’s delta (δ) | Effectiveness levels |
| | |
| | Negligible (N) |
| 0.147 ≤ | Small (S) |
| 0.33 ≤ | Medium (M) |
| 0.474 | Large (L) |

B. ANALYSIS RESULTS OF RQ2

1) EFFECT OF ADVERSARIAL LEARNING

In the objective function of DAFL, we employ the adversarial learning by jointly optimizing the adversarial loss and discrimination loss. To explore the effectiveness of adversarial learning in our approach, we display the values of adversarial loss and discrimination loss from epoch (one epoch = one forward pass and one backward pass of all the training examples) 1 to 10000 in Fig. 3. In Fig. 3, the adversarial loss has almost remained stable (after 10 epochs), while the discrimination loss converges smoothly until about 100 epochs. The results in Fig. 3 are in conformity to the expectation of our approach in CSDP. When the discrimination loss is vibrating, the defect prediction performance will increase. The project discriminator that directs the process of subspace feature discriminate learning is incorporated in the feature transformer. If the value of adversarial loss is soaring, the project discriminator will fail to direct the feature transformer. And if the adversarial loss is optimized to zero, the CPDP will be impossible. The success of project discriminator means the failure of the feature transformer, thus the discriminate feature learning will be ineffective.
TABLE 7. The effectiveness levels of DAFL against each compared method in terms of F-measure and G-measure.

| Measure | Levels | FTcF-MDS | LDS | NSGLP | RusTri | MxTrA+ | CKSDL | ManualDown | CamargoCruz09 | NN-filter | SSTCA +ISDA |
|---------|--------|----------|-----|--------|--------|--------|-------|------------|--------------|-----------|------------|
| F-measure | N | 0 | 0 | 1 | 0 | 2 | 0 | 5 | 3 | 0 | 0 |
|         | S | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|         | M | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|         | L | 16 | 16 | 16 | 16 | 16 | 16 | 11 | 13 | 16 | 16 |
| G-measure | N | 2 | 3 | 5 | 5 | 6 | 6 | 8 | 5 | 3 | 3 |
|         | S | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
|         | M | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|         | L | 14 | 13 | 11 | 10 | 11 | 10 | 8 | 10 | 12 | 12 |

TABLE 8. F-measure and G-measure results using DAFL with \( L_{fd} \), DAFL with \( L_{str} \) and DAFL for each target project.

| Dataset | Target project | DAFL\(_{L_{fd}}\) | DAFL\(_{L_{str}}\) | DAFL | DAFL\(_{L_{fd}}\) | DAFL\(_{L_{str}}\) | DAFL |
|---------|----------------|----------------|----------------|-----|----------------|----------------|-----|
| AEEHM   | EQ            | 0.5402         | 0.5461         | 0.6204 | 0.5942         | 0.5573         | 0.6572 |
|         | JD            | 0.6530         | 0.5978         | 0.6954 | 0.6687         | 0.6083         | 0.7296 |
|         | LC            | 0.7291         | 0.7390         | 0.7455 | 0.7071         | 0.7216         | 0.7336 |
|         | ML            | 0.7698         | 0.7614         | 0.7715 | 0.7357         | 0.7197         | 0.7547 |
|         | PDE           | 0.5722         | 0.6108         | 0.7034 | 0.5447         | 0.5944         | 0.6863 |
| NASA    | CM1           | 0.6803         | 0.7495         | 0.7513 | 0.6765         | 0.7082         | 0.7392 |
|         | MW1           | 0.7820         | 0.7952         | 0.8057 | 0.7482         | 0.7780         | 0.7855 |
|         | PC1           | 0.7145         | 0.7075         | 0.7823 | 0.6910         | 0.7134         | 0.7689 |
|         | PC3           | 0.6547         | 0.6472         | 0.6717 | 0.6479         | 0.6297         | 0.6814 |
|         | PC4           | 0.6754         | 0.6460         | 0.6828 | 0.5780         | 0.6023         | 0.6451 |
| PROMISE | Ant v1.7      | 0.4866         | 0.4863         | 0.5038 | 0.5183         | 0.5135         | 0.5282 |
|         | Camel v1.6    | 0.4230         | 0.4936         | 0.5294 | 0.4942         | 0.5123         | 0.5415 |
|         | Poi v3.0      | 0.7892         | 0.7868         | 0.7702 | 0.5308         | 0.5154         | 0.6479 |
|         | Tomcat v6.0   | 0.7450         | 0.7365         | 0.7368 | 0.6711         | 0.7121         | 0.7225 |
|         | Xalan v2.6    | 0.6461         | 0.6388         | 0.6742 | 0.6678         | 0.6617         | 0.6782 |
|         | Xerces v1.3   | 0.7603         | 0.7469         | 0.7706 | 0.7625         | 0.7498         | 0.7639 |
|         | Overall Average | 0.6715     | 0.6786         | 0.7034 | 0.6379         | 0.6486         | 0.6944 |

FIGURE 3. Time cost, Discrimination loss and Adversarial loss for different numbers of epochs.

2) EFFECT OF FEATURE TRANSFORMER (COMBINING PROJECT DISCRIMINATION AND FEATURE STRUCTURE PRESERVATION)
Feature transformer of DAFL consists of two processes: label prediction and feature structure preservation. To evaluate the effectiveness of this combination, we compare DAFL with two baseline approaches of our approach: DAFL with \( L_{fd} \) only (DAFL\(_{L_{fd}}\)), DAFL with \( L_{str} \) only (DAFL\(_{L_{str}}\)). Table 8 shows the F-measure and G-measure values of DAFL and these baselines on three datasets. The results show that both the feature transformer and project discriminator contribute to the defect prediction. And the performance of optimizing the \( L_{fd} \) and \( L_{str} \) simultaneously in the model is better than optimizing only one of them.

3) EFFECT OF THE PSEUDO LABEL
In order to assess the effectiveness of pseudo label, we give the F-measure and G-measure results which we perform experiment without pseudo label. In feature discrimination process, we only discriminate the class from the target labeled instances without the pseudo label from source instances. In Table 9, we can acquire better experiment results with pseudo labels. To make full use of discrimination of unlabeled data, we apply CLCR technology to estimate the labels of unlabeled data from existing label information in data pre-processing, and then use the pseudo labels and true labels in feature discrimination process. The experiment results show that the feature discrimination process will perform better with both pseudo labels and existing labels.
TABLE 9. F-measure and G-measure results using DAFL without pseudo label (DAFL without pl) and DAFL for each target project.

| Target project | F-measure | G-measure |
|----------------|-----------|-----------|
|                | DAFL without pl | DAFL     | DAFL without pl | DAFL     |
| EQ             | 0.5637    | 0.6204   | 0.6018          | 0.6572   |
| JDT            | 0.6553    | 0.6954   | 0.6694          | 0.7296   |
| LC             | 0.7348    | 0.7455   | 0.6788          | 0.7336   |
| ML             | 0.7695    | 0.7715   | 0.7241          | 0.7547   |
| PDE            | 0.5767    | 0.7034   | 0.5521          | 0.6863   |
|               | 0.7066    | 0.7513   | 0.6777          | 0.7892   |
|                | 0.7651    | 0.8057   | 0.7358          | 0.7855   |
| NASA           | 0.7978    | 0.7823   | 0.7780          | 0.7689   |
|                | 0.7233    | 0.6717   | 0.7212          | 0.6814   |
|                | 0.7458    | 0.6828   | 0.7157          | 0.6451   |
| Ant v1.7       | 0.4745    | 0.5038   | 0.5084          | 0.5282   |
| Camel v1.6     | 0.4731    | 0.5294   | 0.5138          | 0.5415   |
| PROMISE        | 0.7235    | 0.7702   | 0.6310          | 0.6479   |
| Tomcat v6.0    | 0.7145    | 0.7368   | 0.6775          | 0.7225   |
| Xalan v2.6     | 0.6337    | 0.6742   | 0.6533          | 0.6782   |
| Xerces v1.3    | 0.7354    | 0.7706   | 0.7379          | 0.7639   |
| Overall Average| 0.6777    | 0.7034   | 0.6637          | 0.6944   |

FIGURE 4. Average G-measure values versus different ratios of labeled instances in the target project. (a) On the AEEEM dataset. (b) On the NASA dataset. (c) On the PROMISE dataset.

C. DISCUSSION

1) INFLUENCE OF LABELED DATA SIZE
In this section, we investigate the effect of different ratios of labeled instances in target project to DAFL. The percentage of labeled instances in the target project is range from 10% to 30%. For a specific percentage of labeled instances, we report the mean results of each method on different target projects. We report the average G-measure results on AEEEM, NASA, and PROMISE datasets in Fig. 4.(a)-(c). In Fig. 4, when the ratio of labeled instances increases, the prediction performance of all the methods improves. And DAFL always outperforms most of the baseline methods.

2) HOW WE SET THE NUMBER OF EPOCH?
Deciding the number of epochs is an important step for training the model. In general, the higher the number of epochs, the more the time cost. We choose Tomcat project as target project and camel as source project to conduct the experiment. We use a machine with 32GB RAM, Intel Core i7-8700k processor and an Nvidia GeForce GTX 1080Ti GPU card. We use discriminate loss $L_D$ and adversarial loss $L_D$ to evaluate the parameter. The values of epochs range from 1 to 10000. In Fig. 3, when we set the number of epochs to 100, the values of discriminate loss and adversarial loss are stable, and the time we cost is little. In this study, the number of epochs is set to 100 and the time cost is only 60 seconds.

3) PARAMETER SETTING OF THE PARAMETER $\beta$
As mentioned in Section IV-F, we set $\beta$ value to 0.01 to 100. In this section, we discuss the influences of different values of $\beta$. We set the range of the value as [0.01,100] on AEEEM dataset. Fig. 5 depicts the F-measure and G-measure values of DAFL with different $\beta$ values on AEEEM dataset. From Fig. 5, when $\beta = 0.1$, DAFL achieves higher F-measure and G-measure values, but the range of value differences is small.

VI. THREATS TO VALIDITY

A. THREATS TO EXTERNAL VALIDITY
In this paper, although 16 projects from three datasets and various metrics are used in experiment, e.g., complexity,
code size. We cannot be sure whether DAFL can be applied to other systems and metrics.

B. THREATS TO INTERNAL VALIDITY
We implement the baselines by carefully following the original papers. These compared related works do not provide the source codes of their works except for CKSDL, although we try our best to guarantee the accuracy of the implementation, our implement may not completely consistent with original paper.

C. THREATS TO CONSTRUCT VALIDITY
We mainly used F-measure and G-measure, which have been widely used to evaluate the effectiveness of defect prediction models, to evaluate the prediction performance. Although these two measures are widely used in most compared methods, some biases still exist.

VII. CONCLUSION
In this work, we proposed DAFL approach, a novel approach to learn both discriminative and correlative representations in common subspace for CSDP. DAFL takes advantage of adversarial learning that performs a minimax game into two adversarial stages: a feature transformer generates intra-class correlated and inter-class discriminated representations, a project discriminator tries to discriminate the project attribute of an unknown representation to minimize the distribution difference between source project and target project data. In feature transformer, we designed triplet constraints to ensure the feature structure is preserved, which ensures the feature representations learned are intra-class correlated and inter-class discriminated in common subspace. We construct experiment on three widely used defect datasets and the result analyses have demonstrated the effectiveness and efficiency of our DAFL.

For future work, we will adjust DAFL framework to better deal with more complex datasets including both commercial proprietary closed and open source projects.

REFERENCES
[1] N. E. Fenton and M. Neil, “A critique of software defect prediction models,” IEEE Trans. Softw. Eng., vol. 25, no. 5, pp. 675–689, Sep. 1999.
[2] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, “Benchmarking classification models for software defect detection: A proposed framework and novel findings,” IEEE Trans. Softw. Eng., vol. 34, no. 4, pp. 485–496, Jul. 2008.
[3] Y. U. B. Challagulla, F. B. Bastani, I.-L. Yen, and R. A. Paul, “Empirical assessment of machine learning based software defect prediction techniques,” Int. J. Artif. Intell. Tools, vol. 17, no. 2, pp. 389–400, Apr. 2008.
[4] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, “A systematic literature review on fault prediction performance in software engineering,” IEEE Trans. Softw. Eng., vol. 38, no. 6, pp. 1276–1304, Nov. 2012.
[5] S. Wang and X. Yao, “Using class imbalance learning for software defect prediction,” IEEE Trans. Rel., vol. 62, no. 2, pp. 434–443, Jun. 2013.
[6] A. Okutan and O. T. Yildiz, “Software defect prediction using Bayesian networks,” Empirical Softw. Eng., vol. 19, no. 1, pp. 154–181, Feb. 2014.
[7] Y. Zhang, D. Lo, X. Xia, and J. Sun, “An empirical study of classifier combination for cross-project defect prediction,” in Proc. COMPSAC, Jul. 2015, pp. 264–269.
[8] Y. Zhou, Y. Yang, H. Lu, L. Chen, Y. Li, Y. Zhao, J. Qian, and B. Xu, “How far we have progressed in the journey? An examination of cross-project defect prediction,” ACM Trans. Softw. Eng. Methodol., vol. 27, no. 1, pp. 1–51, Apr. 2018.
[9] C. Catal and B. Diri, “Unlabelled extra data do not always mean extra performance for semi-supervised fault prediction,” Expert Syst., vol. 26, no. 5, pp. 458–471, Nov. 2009.
[10] Y. Jiang, M. Li, and Z.-H. Zhou, “Software defect detection with Rocus,” J. Comput. Sci. Technol., vol. 26, no. 2, pp. 328–342, Mar. 2011.
[11] H. Lu, B. Cukic, and M. Culp, “An iterative semi-supervised approach to software fault prediction,” in Proc. PROMISE, 2011, pp. 1–10.
[12] S. Huda, S. Alyahya, M. Mohsin Ali, S. Ahmad, J. Abawajy, H. Al-Dossari, and J. Yearwood, “A framework for software defect detection and metric selection,” IEEE Access, vol. 6, pp. 2844–2858, 2018.
[13] E. A. Felix and S. P. Lee, “Integrated approach to software defect prediction,” IEEE Access, vol. 5, pp. 21524–21547, 2017.
[14] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, “Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning,” in Proc. ESEC/FSE, 2015, pp. 496–507.
[15] Z. Li, X.-Y. Jing, F. Wu, X. Zhu, B. Xu, and S. Ying, “Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction,” Autom. Softw. Eng., vol. 25, no. 2, pp. 201–245, Jun. 2018.
[16] J. Zheng, “Cost-sensitive boosting neural networks for software defect prediction,” Expert Syst. Appl., vol. 37, no. 6, pp. 4537–4543, Jun. 2010.
[17] Ö. F. Arar and K. Ayan, “Software defect prediction using cost-sensitive neural network,” Appl. Soft Comput., vol. 33, pp. 263–277, Aug. 2015.
[18] D. Li, P. He, J. Zhu, and M. R. Lyu, “Software defect prediction via convolutional neural network,” in Proc. QRS, Jul. 2017, pp. 318–328.
[19] S. Wang, T. Liu, and L. Tan, “Automatically learning semantic features for defect prediction,” in Proc. ICSE, 2016, pp. 297–308.
[20] Y. Kim and B. Toomajian, “Hand gesture recognition using micro-Doppler signatures with convolutional neural network,” IEEE Access, vol. 4, pp. 7125–7130, 2016.
[21] C. Yin, Y. Zhu, J. Fei, and X. He, “A deep learning approach for intrusion detection using recurrent neural networks,” IEEE Access, vol. 5, pp. 21954–21961, 2017.
[22] S. Wang, T. Liu, J. Nam, and L. Tan, “Deep semantic feature learning for software defect prediction,” IEEE Trans. Softw. Eng., to be published.
[23] Z. Xu, P. Yuan, T. Zhang, Y. Tang, S. Li, and Z. Xia, “HDA: Cross-project defect prediction via heterogeneous domain adaptation with dictionary learning,” IEEE Access, vol. 6, pp. 57597–57613, 2018.
[24] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, “Cross-project defect prediction: A large scale experiment on data vs. domain vs. process,” in Proc. ESEC/FSE, 2009, pp. 91–100.
[25] Y. Ma, G. Luo, X. Zeng, and A. Chen, “Transfer learning for cross-company software defect prediction,” Inf. Softw. Technol., vol. 54, no. 3, pp. 248–256, Mar. 2012.
[26] J. Nam, S. J. Pan, and S. Kim, “Transfer defect learning,” in Proc. ICSE, May 2013, pp. 382–391.
[27] L. Deng and D. Yu, “Deep learning: Methods and applications,” Found. Trends Signal Process., vol. 7, nos. 3–4, pp. 197–387, Jun. 2014.
[28] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, p. 436, 2015.
XIAO-YUAN JING received the Ph.D. degree in pattern recognition and intelligent system from the Nanjing University of Science and Technology, in 1998. He was a Professor with the Department of Computer Science, Shenzhen Research Student School, Harbin Institute of Technology, in 2005. He is currently a Professor with the School of Automation, Nanjing University of Posts and Telecommunications, China, the School of Computer, Wuhan University, and the School of Computer, Guangdong University of Petrochemical Technology. He has published over 100 scientific articles in international journals and conferences, including TIP, TIFS, TCB, TCSVT, TMM, TR, TSMC-B, CVPR, AAAI, IJCAI, and ICME. His research interests include pattern recognition, image processing, computer vision, and machine learning.

FEI WU received the Ph.D. degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2016. His research interests include pattern recognition, artificial intelligence, and software engineering.

JUANJUAN LI is currently pursuing the M.S. degree with the School of Automation, Nanjing University of Posts and Telecommunications, Nanjing, China. Her research interest includes software engineer.

DANLEI XING is currently pursuing the M.S. degree with the School of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China. His research interest includes software engineer.

HAOWEN CHEN is currently pursuing the Ph.D. degree with the School of Computer Science, Wuhan University, Wuhan, China. His research interests include software defect prediction and machine learning.

YANFEI SUN received the Ph.D. degree in information network from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2006. He has been a Professor with the School of Automation, Nanjing University of Posts and Telecommunications, since 2006. His current research interests include intelligent optimization, network management, machine learning, and future networks.