Hindawi
IET Software
Volume 2024, Article ID 1561351, 19 pages
https://doi.org/10.1049/2024/1561351

Research Article

Balanced Adversarial Tight Matching for Cross-Project Defect Prediction

Siyu Jiang[1], **Jiapeng Zhang**[1], **Feng Guo**[1], **Teng Ouyang**[1], **and Jing Li**[2]

[1]*School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou, China*
[2]*Guangzhou City University of Technology, Guangzhou, China*

Correspondence should be addressed to Jing Li; lijing1@gcu.edu.cn

Received 13 February 2024; Revised 15 April 2024; Accepted 2 May 2024; Published 16 May 2024

Academic Editor: Manuel Angel Serrano

Cross-project defect prediction (CPDP) is an attractive research area in software testing. It identifies defects in projects with limited labeled data (target projects) by utilizing predictive models from data-rich projects (source projects). Existing CPDP methods based on transfer learning mainly rely on the assumption of a unimodal distribution and consider the case where the feature distribution has one obvious peak. However, in actual situations, the feature distribution of project samples often exhibits multiple peaks that cannot be ignored. It manifests as a multimodal distribution, making it challenging to align distributions between different projects. To address this issue, we propose a balanced adversarial tight-matching model for CPDP. Specifically, this method employs multilinear conditioning to obtain the cross-covariance of both features and classifier predictions, capturing the multimodal distribution of the feature. When reducing the captured multimodal distribution differences, pseudo-labels are needed, but pseudo-labels have uncertainty. Therefore, we additionally add an auxiliary classifier and attempt to generate pseudo-labels using a pseudo-label strategy with less uncertainty. Finally, the feature generator and two classifiers undergo adversarial training to align the multimodal distributions of different projects. This method outperforms the state-of-the-art CPDP model used on the benchmark dataset.

1. Introduction

Software defect prediction (SDP) [1] aims to enhance the efficiency of resource allocation by predicting program modules that are likely to have defects before the software enters the testing phase. This allows for a targeted allocation of resources for code inspection based on the prediction outcomes. SDP commonly relies on historical project data to construct predictive models. However, in the initial stages of developing a new project, historical data are often scarce.

To tackle this challenge, researchers [2, 3] proposed cross-project defect prediction (CPDP) technology. This technology makes use of defect data from projects originating from different sources, enabling the application of SDP in the early phases of new projects. In CPDP, a prominent issue arises from the differing sources of various projects, resulting in distribution disparities. If the classification boundary learnt from the source project probability distribution is applied directly to the target project, it may underperform on the target project. This is primarily due to the distinct probability distributions between the source and target projects. Applying the knowledge learned from the probability distribution of the source project directly to the target project can significantly degrade the prediction performance [4].

To mitigate this, researchers have suggested employing transfer learning methods. These methods aim to reduce the distribution disparity between different projects, thereby addressing the issue of disparate distributions. For instance, Qiu et al. [5] utilized both semantic features and handcrafted features as joint features [6]. Simultaneously, they incorporate the maximum mean discrepancy (MMD) [7] to minimize the feature distribution gap between the source and target project. Similarly, Xing et al. [8] propounded an adversarial long short-term memory neural network (G-LSTM), incorporating joint features and employing adversarial training to mitigate distribution disparities. These methods take into account the reduction of differences in the distribution of features between source and target items, which is

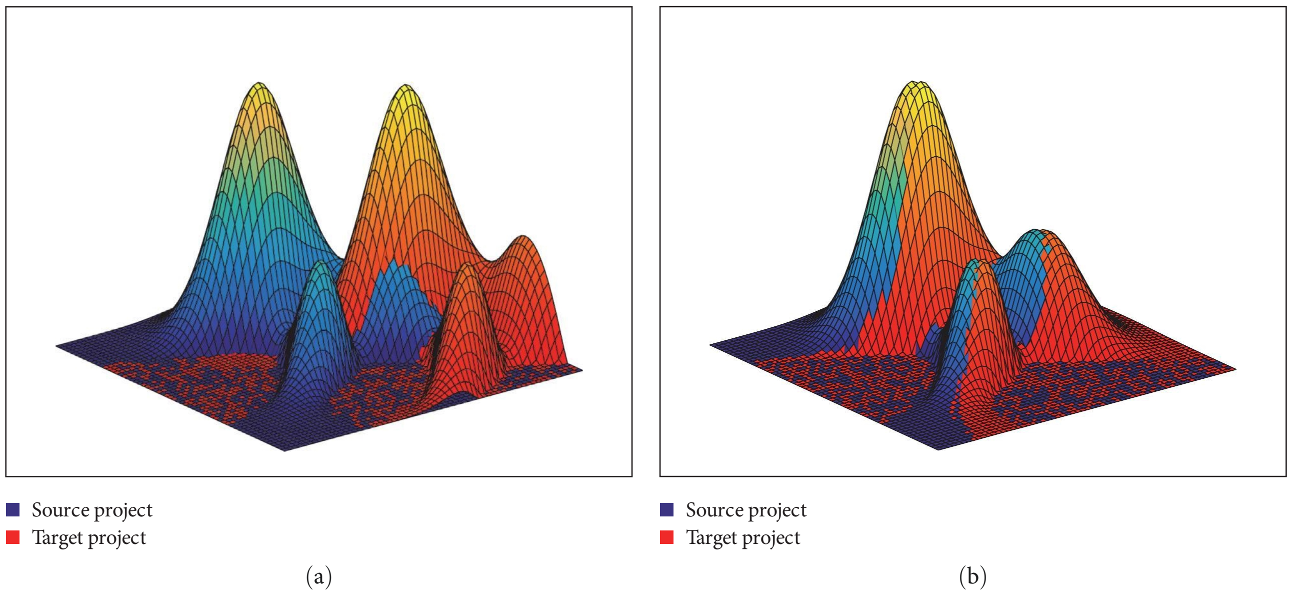


Figure 1: Schematic representation of the multimodal distribution of software project features: (a) shows the effect of other researchers' methods, only considering the alignment of single peaks; (b) shows the effect of our method, which achieves multipeak alignment by capturing multiple peaks.

essential to improve the effectiveness of CPDP. They [5, 8] attempt distribution alignment based on the assumption that the features of the project have a unimodal distribution. However, in practice, software projects are often codeveloped by multiple developers, each with their own coding style. Additionally, modules with different functions vary greatly in complexity. Particularly, handcrafted features are typically extracted based on software metrics, code analysis, and other methods, which have different statistical properties compared to those extracted by deep learning. These differences result in the feature distribution of the project not exhibiting a single ideal peak but rather multiple distinct peaks. In other words, project features actually display a multimodal distribution. Each of these peaks typically represents a pattern or cluster within the project, and the different peaks indicate a higher concentration of the frequency or value of the feature in different regions [9]. Traditional methods can only align a portion of the peaks and fail to achieve complete alignment (as illustrated in Figure 1(a)). This still results in significant distribution disparities, subsequently diminishing the accuracy of the cross-project predictor.

To address the challenge, we propose the balanced adversarial tight matching (BATM) method. For program features extracted from projects, we obtain the cross-covariance between features and classifier predictions using multilinear conditioning to capture the multimodal distribution of features. Based on the captured distribution, we need to further know the distribution differences between projects so as to reduce it. Therefore, we measure the distribution differences between different projects by extending the maximum density divergence (MDD). When calculating MDD, the label information is needed, but the target project does not have the label information, so we introduced pseudo-labels. Due to the distribution gap between projects, there is a certain deviation between the pseudo-labels and the real labels. To solve this problem, we additionally add an auxiliary classifier to correct the pseudo-labels by minimizing the prediction bias between the main classifier and the auxiliary classifier. The intuitive effect of the BATM method is shown in Figure 1(b). It continuously "pulls" unaligned peaks over during training and eventually achieves the alignment of multimodal distributions. Additionally, we conducted specific experimental verification in Section 6.3.

This paper's primary contributions are outlined below:

(i) We propose an adversarial defect prediction framework based on multilinear conditioning. It achieves a closer alignment between the distributions of different projects by capturing the multimodal distribution of data.

(ii) There will inevitably be noise in pseudo-labels, leading to uncertainty. We reduce pseudo-label uncertainty by minimizing the prediction bias. Specifically, it acts as a regularization term to correct the learning of noisy pseudo-labels, making the MDD measure of distribution differences more reliable and effective.

(iii) We conducted comparative experiments between our method and 9 baseline methods on 11 common open-source projects. The experimental results show that our method achieves state-of-the-art results in F_1 measure, balanced accuracy, and AUC.

The residual content of this paper is relevant in the following: Section 2 reports the related work. Section 3 narrates the specifics of our proposed method. Sections 4 and 5 cover the experimental setup and present the experimental results. Section 6 presents the experimental discussion. Finally, Section 7 offers a summary of our contributions.

2. Related Work

In CPDP, several methods have been proposed to exploit the features of the program. Nguyen and Nguyen [10] propounded a source code semantic model called abstract syntax trees (ASTs). Further, based on AST, Wang et al. [6] employed a deep Bayes network (DBN) model to acquire semantic features from ASTs, bridging the semantic gap between different project features and defect prediction. Inspired by the ability of convolutional neural network (CNN) to effectively extract semantic features from text [11], Li et al. [12] investigated the application of CNN for generating semantic features of programs. Recognizing the significance of code context semantics, Xing et al. [8] presented G-LSTM learning model to autonomously extract program semantics and context-dependent features.

To enhance prediction across diverse projects, researchers have explored transfer learning methods to align feature distributions between source and target projects. Ma et al. [13] propounded the transfer naive Bayes (TNB) model to address the CPDP problem. Pan et al. [14] investigated the use of transfer component analysis (TCA), a transfer learning technology, in the context of CPDP. TCA involves mapping the source dataset and the target dataset into a latent space, facilitating data migration by minimizing the distance between them. Subsequently, they proposed an enhanced transfer defect learning method called TCA+ [15], which employs customized normalized extended TCA. The findings revealed that TCA+ significantly outperformed other comparative models in certain projects in terms of performance. Given that joint features can more effectively capture the distinctive features of project defects, Zou et al. [16] devised a joint feature representation learning method. They implemented a repeated pseudo-label strategy to narrow down the feature distribution gap among different projects. In order to extract transferable features, Qiu et al. [5] propounded a transfer convolutional neural network (TCNN) model. They added a matching layer based on CNN for feature mapping and utilized MMD [7] to mitigate the differences between various projects. Additionally, Huang et al. [17] utilized serialized ASTs and handcrafted feature generation vectors, employing multicore MMD to align the feature distribution of different projects. Given the substantial imbalance in class within the project, Tang et al. [18] presented an algorithm for transfer learning, named TsboostDF. This algorithm specifically addresses both the knowledge transfer and class imbalance challenges inherent in CPDP. Recognizing the effectiveness of adversarial generation, Cheng et al. [19] applied kernel-based principal component analysis to transform instances, mapping them to a high-dimensional feature space. Subsequently, they employed adversarial learning to acquire representative features for model construction. Song et al. [20] employed two classifiers for detecting target samples that are distant from the source sample support vector through adversarial training. They also utilized a generator to minimize the difference in the target samples, aligning the distribution.

However, the previously mentioned CPDP methods all operate under the assumption of unimodal distribution. It may potentially hinder performance improvement as they overlook the presence of multimodal data distributions. In this paper, we propose a novel BATM model. It not only considers the multimodal distribution but also enhances the effectiveness of MDD using a pseudo-label marking method with lower uncertainty. This approach leads to a more significant improvement in performance.

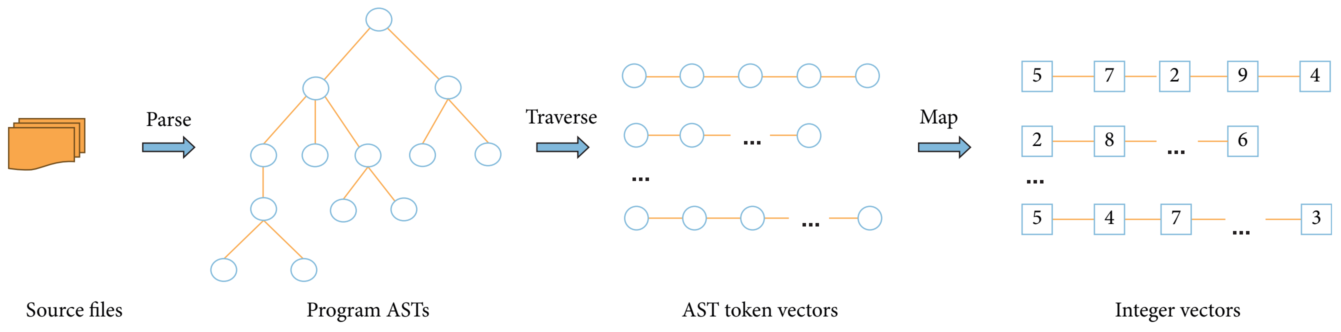
3. Methodology

In this section, we initially provide a formal description of the CPDP problem. Then, the overall process of the BATM method is presented, and the specific modules of the method are reported in order.

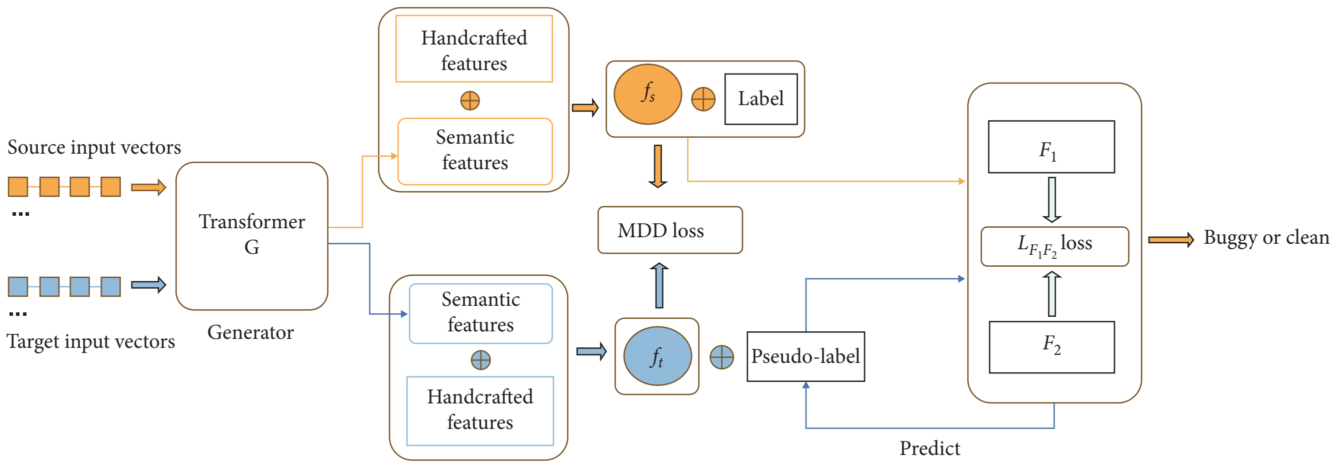
3.1. Problem Definition. This paper defines the source project as the source domain $\mathfrak{D}_S$, comprising the feature space X_S of the source project data along with its corresponding probability distribution P. The sample set from the source domain is denoted as $X_S = \{x_{S_1}, \ldots, x_{S_n}\} \in X_S$. Similarly, the target project is considered the target domain $\mathfrak{D}_T$, consisting of the feature space X_T of the target project data and its corresponding probability distribution Q. The sample set from the target domain is represented as $X_T = \{x_{T_1}, \ldots, x_{T_n}\} \in X_T$.

Typically, while two software projects from different sources may share some common metric elements, the distributions to which these metric elements adhere are generally distinct, i.e., $P \neq Q$. The objective of CPDP is to train a classifier dependent on the data from the source project and apply it to the target project. To achieve this cross-domain migration between two projects from different sources, it is often necessary to minimize the distribution disparity between the source project and the target project, a process known as domain adaptation. In the model training process, CPDP researchers also utilize some unlabeled target domain data as auxiliary information to construct the final prediction model, achieving more effective transfer results.

3.2. Overall Workflow. The overall workflow of our proposed CPDP method is depicted in Figure 2. Specifically, our framework consists of the following steps: We begin by parsing the project source file into an AST and traversing the AST to acquire the token vector. The token vector is then transformed into an integer vector through predefined mapping rules. Next, the integer vector is input into the transformer, which acts as a generator to extract semantic features. These semantic features are subsequently combined with handcrafted features to form joint features. After data preprocessing, we first train a main classifier F_1 and an auxiliary classifier F_2 based on the source project and then use the main classifier F_1 to generate pseudo-labels for the target project. Next, the joint features and classifier predictions are performed in multilinear conditioning in order to capture the multimodal distribution of the data. Subsequently, we utilize MDD to train a generator and two classifiers based on the adversarial learning method. In this process, the pseudo-labels are continuously corrected, and the purpose of reducing the multimodal distribution difference between the two projects is achieved. Finally, the constructed model is deployed on the target project to predict instances where defects may



(a)



(b)

Figure 2: Overall workflow: (a) project data preprocessing; (b) defect prediction model construction.

occur. Below, we provide details about the algorithm of our proposed method.

3.3. Generating Input Vectors. The AST is a semantic structure extracted from the source code, represented as a multi-branch tree. It serves to depict the syntax structure of the source code. The AST offers a high level of information compression and comprises a limited set of node types. Its structure aligns well with the semantic information of the program code. These advantages enable ASTs to be initially used in defect prediction [6].

In Figure 2(a), we employ the open-source compilation tool Javalang to process Java source files and create the corresponding AST. While the AST contains various types of nodes, a significant portion of them are redundant, with only a small portion highly related to code defects. Building on prior research [6], we thoughtfully select four types of nodes: method invocation, declaration, control flow, and other essential node types. Then, the AST is traversed through depth-first after the filtered node is obtained to obtain the mark sequence. As these tag sequences are string representations and cannot be directly fed into the transformer network, we establish a one-to-one mapping rule between nodes and positive integers. The mapping enables the node label sequence to be transformed into the corresponding integer vector, which can then serve as input for the transformer network.

3.4. Generator. Considering that the source code of a software project is also a special form of text, it follows certain grammatical rules, with keywords, etc., holding specific semantic meanings. Therefore, CPDP tasks share significantly related to natural language processing (NLP) tasks [21, 22]. The transformer network [23], known for its proficiency in learning long-term dependencies through complex computations, is especially well-suited for handling NLP tasks. We argue that the transformer network, with its self-attention mechanism, is effective at extracting contextual and semantic features from software projects. Therefore, the BATM method utilizes this technique to extract semantic features from the program, serving as a feature generator for the entire model.

The structural diagram of the transformer is illustrated in Figure 3. In the initial step, position encoding is applied to each integer vector within the input sequence data X. In this manner, words at different positions will be assigned varying importance as they pass through the self-attention mechanism. This enables the model to comprehend the order of words in the input sequence. Here, we utilize $\mathrm{PE}_{(\mathrm{pos},2i)}$ and $\mathrm{PE}_{(\mathrm{pos},2i+1)}$ to represent the position encoding at indices $2i$ and $2i+1$ of the given positioning pos in the sequence, respectively. The corresponding formula is described as follows:

$$\begin{aligned}\mathrm{PE}_{(\mathrm{pos},2i)} &= \sin\left(\mathrm{pos}/10{,}000^{2i/d_{\mathrm{model}}}\right)\\ \mathrm{PE}_{(\mathrm{pos},2i+1)} &= \cos\left(\mathrm{pos}/10{,}000^{2i/d_{\mathrm{model}}}\right).\end{aligned}\tag{1}$$

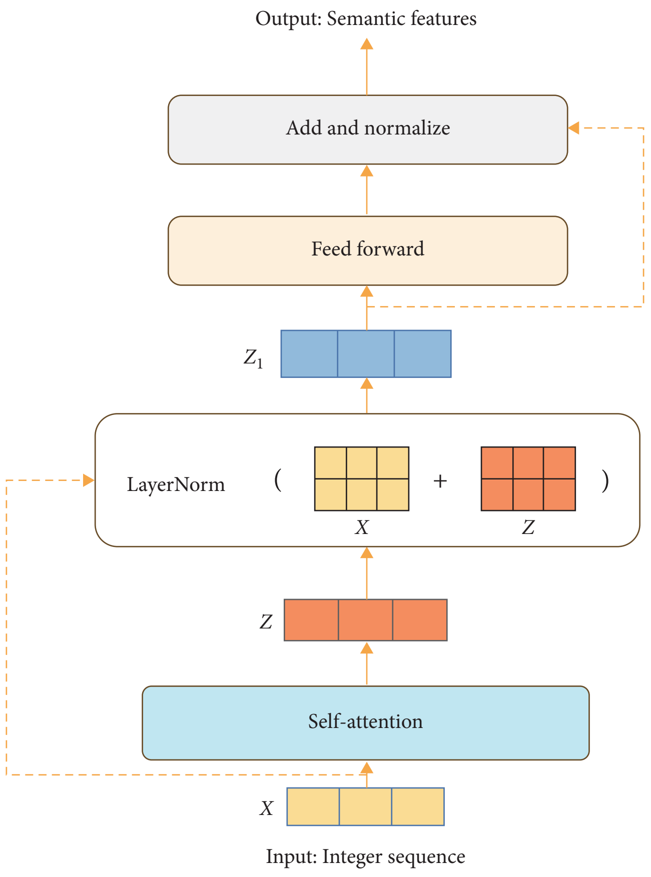


FIGURE 3: The transformer model architecture.

Among them, d_{model} represents the embedding dimension of the model, which is the dimension of word embedding. The constant 10,000 is used to adjust the periodicity of the sine function sin and the cosine function cos.

Subsequently, the integer vector with added position encoding undergoes processing in the self-attention layer. The weight matrices $W_Q \backslash W_K \backslash W_V$ learned by the three models are multiplied by the i element x_i in the input sequence to obtain the $Q \backslash K \backslash V$ vectors. Then, calculate their self-attention values as shown in Formula (2):

$$\text{Self-attention}\,(Q, K, V) = \text{soft max}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{2}$$

In the formula, QK^{T} represents the inner product of the transpose of the query vector Q and the key vector K. $\sqrt{d_k}$ is a normalization factor, typically the square root of the dimension of the K matrix. This normalization operation ensures that the input dimensionality is not affected in the attention calculation. The softmax operation is employed to normalize the attention score into a probability distribution. Finally, when multiplied by the attention weight V, the self-attention output is obtained. As depicted in Figure 3, Z represents the resulting self-attention output.

Following this, the output of the self-attention layer performs standardization through the layer normalization layer to ensure model stability. In the feed-forward layer, features are mapped to a higher-dimensional space using an activation function. This increases the model's representational capacity and enables it to capture richer semantic information. It serves to further process the normalized representation for extracting higher-level semantic features. Finally, normalization occurs through the add and normalize layer to ensure a stable model output.

To enhance feature transferability and separability, we combine the semantic features acquired by the transformer with traditional handcrafted features to form joint features.

3.5. Adversarial Training. The adversarial training objective of the BATM model is to effectively capture the multimodal distribution of data through multilinear conditioning. This is achieved by integrating it with MDD to provide a more accurate measure of distribution disparities between different projects. Ultimately, this process aligns the distributions and enhances the prediction accuracy.

Initially, the model trains the generator G and classifiers F_1 and F_2 using the source projects. Due to the ample labeled samples available for source projects, the classifier is able to classify them correctly after training.

In detail, BATM engages in adversarial training on the generator and classifier using source project samples. This ensures that the model can effectively classify the source project samples. This article chooses the logistic regression (LR) classifier as the foundational classifier. Given that the CPDP problem entails a binary classification task, we employ binary cross-entropy loss in this step, as depicted below:

$$\begin{aligned} L_{\text{bce}} &= -E[y_S \log p(y|x_S) + (1 - y_S)\log(1 - p(y|x_S))] \\ p(y|x_S) &= \frac{p_1(y|x_S) + p_2(y|x_S)}{2}. \end{aligned} \tag{3}$$

Here, $p_1(y|x_S)$ and $p_2(y|x_S)$ denote the outputs of classifiers F_1 and F_2, respectively. Unless stated otherwise, the classifier outputs refer to the average result produced by both classifiers F_1 and F_2.

Subsequently, the trained classifier is employed to classify the target projects and assign corresponding pseudo-labels. Next, multilinear conditioning comes into play to grasp the multimodal distribution of sample features. Considering that uncertainty in discriminative information can introduce prediction noise, we aim to minimize this uncertainty by reducing the prediction bias between two classifiers.

In this paper, multilinear conditioning is achieved through the use of a multilinear map. A multilinear map is determined by the outer product of numerous random vectors [24]. For two random vectors x and y, the joint distribution $P(x, y)$ can be represented by the cross-covariance $E_{xy}[\varphi(x) \otimes \varphi(y)]$, where φ is the feature map generated by a specific reproducing kernel [25, 26]. Suppose there are n feature vectors $x_i, i \in [1, n]$, their joint feature vector x, label vector y, then the formal representation of the cross-covariance $E_{xy}[\varphi(x) \otimes \varphi(y)]$ is as follows:

$$E_{xy}[\varphi(x) \otimes \varphi(y)] = E_{x_1 y}[\varphi(x_1) \otimes \varphi(y)] \oplus \ldots \oplus E_{x_n y}[\varphi(x_n) \otimes \varphi(y)]. \tag{4}$$

Therefore, the multilinear map $x \otimes y$ can capture the cross-covariance between feature representations and labels, thereby capturing the multimodal distribution of complex features. The representation of the multilinear map $M_\otimes$ of features and predicted labels is as follows:

$$M_\otimes(f, g) = f \otimes g. \tag{5}$$

Among the multilinear map $M_\otimes$, two notable ones are f and g. f represents the feature representation, which combines the joint features of the source project and the target project. Meanwhile, g represents the average value predicted by classifiers F_1 and F_2. However, one drawback of a multilinear map is dimensionality explosion. Here, let d_f and d_g denote the dimensions of vectors f and g, respectively. The dimension of the multilinear map $f \otimes g$ is given by $d_f \times d_g$. If this dimension becomes too high, parameter explosion will occur when embedding a deep network. To tackle this problem, the paper suggests a resolution using the random multilinear map to mitigate the dimension explosion. Since the inner product on $M_\otimes$ can be accurately estimated by the inner product on $M_\odot$ [27], in order to enhance the computational efficiency, we use $M_\odot(f, g)$ to represent a random multilinear map. Its formulaic description is as follows:

$$M_\odot(f, g) = \frac{1}{\sqrt{d}}(R_f f) \odot \left(R_g g\right), \tag{6}$$

where $\odot$ is the element-wise product, dimension $d \ll d_f \times d_g$, R_f, and R_g are random matrices sampled only once and fixed in training. Each element R_{ij} follows a symmetric distribution with single variance, that is $E[R_{ij}] = 0, E[\mathrm{R}_{ij}^2] = 1$.

Furthermore, we require a metric to quantify this distribution disparity, which is addressed by MDD in this context. MDD [28] serves a dual purpose: not only does it gauge the distribution difference between two distinct multimodal distributions, but it also diminishes the divergence between domains. This leads to an increase in intraclass density, rendering the discriminative information of defective and nondefective samples on the target project more discernible. Thereby, it benefits the classifier's prediction of the target project. We denote the difference between P and Q as $\mathrm{MDD}(P, Q)$ and aim to harmonize the source and target domain distributions by minimizing $\mathrm{MDD}(P, Q)$. The formal description of MDD is as follows:

$$\begin{aligned}\mathrm{MDD}(P, Q) = {} & E_{X_s \sim P, X_t \sim Q}[| |X_s - X_t| |_2^2] \\ & + E_{X_s, X_s' \sim P}[| |X_s - X_s'| |_2^2] \\ & + E_{X_t, X_t' \sim Q}[| |X_t - X_t'| |_2^2].\end{aligned} \tag{7}$$

Among these, X_s' and X_t' represent independent and identically distributed copies of X_t and X_t, respectively. In the first term of MDD, we utilize the squared Euclidean norm distance to minimize the distribution difference between P and Q. The subsequent two terms serve to maximize the within-project density of P and Q, respectively. This method not only aligns the distribution differences between the source and target projects but also brings the two projects themselves closer together. In practice, we calculate the MDD loss by the following formula:

$$\begin{aligned}L_{\mathrm{mdd}} = {} & \frac{1}{n_b}\sum_i^{n_b}\|f_{s,i} - f_{t,i}\|_2^2 + \frac{1}{m_s}\sum_{y_{s,i}=y_{s,j}'}\left\|f_{s,i} - f_{s,j}'\right\|_2^2 \\ & + \frac{1}{m_t}\sum_{y_{t,i}=y_{t,j}'}\left\|f_{t,i} - f_{t,j}'\right\|_2^2.\end{aligned} \tag{8}$$

In this context, f represents the generated sample feature, n_b denotes half of the batch size. Additionally, m_s and m_t, respectively, signify the number of categories in the source domain and target domain within the batch. These quantities can be determined by examining the label of each instance in the present batch. $y_{s,i} = y_{s,j}'$ represents $x_{s,i}$ and $x_{s,i}'$ have the same label, where $x_{s,i}$ represents the i sample of the source project, $x_{s,i}'$ is a copy of $x_{s,i}$.

From the above description of MDD, we can observe that one of its objectives is to maximize intraclass density. To achieve this goal, the label information of the project samples must be utilized. In the case of the target project, obtaining its label information is not feasible, so pseudo-labels are employed instead. However, it is important to note that the pseudo-labels assigned to target project samples are uncertain, as these labels may contain prediction noise [29]. While most pseudo-labels are accurate, some incorrect labels might be present. Fine-tuning the model on noisy labels could transfer the error to the adapted model. This transfer may potentially exacerbate the adaptation process, especially for samples with uncertain predictions [30]. The auxiliary classifier F_2 added in this paper differs from the classifier F_1 solely in terms of the initialization method. Both are capable of classifying project samples. By minimizing the prediction deviation between the two classifiers, target projects can be more accurately labeled with corresponding pseudo-labels in the training process.

To account for label noise, we measure pseudo-label uncertainty by prediction bias. In practice, we utilize the KL divergence D_{KL} predicted by classifiers F_1 and F_2 as an estimate of the bias as follows:

$$D_{\mathrm{kl}} = E\left[F_1\left(x_t^j|\theta_t\right)\log\left(\frac{F_1\left(x_t^j|\theta_t\right)}{F_2\left(x_t^j|\theta_t\right)}\right)\right]. \tag{9}$$

In this context, $F_1(x_t^j|\theta_t)$ signifies the output of the main classifier F_1, while $F_2(x_t^j|\theta_t)$ denotes the output of the auxiliary classifier F_2. When these two classifiers offer divergent class predictions, the prediction deviation will yield a substantial value. This highlights the model's uncertainty in its predictions. We attempt to reduce this prediction bias to

```
Input: Source project and label data $X_S = \{x_{S_i}\}_{i=1}^n$, $Y_S = \{y_{S_i}\}_{i=1}^n$, target project data $X_T = \{x_{T_i}\}_{i=1}^m$, parameters of classifiers $F_1$ and $F_2$ and generators $\theta_{F_1}, \theta_{F_2}$, and $\theta_G$, learning rate $\alpha$, pseudo-label $\rho$, output of the generator to the target project $p(y|x_T)$, maximum iteration $T$.
Ouput: $\theta_{F_1}, \theta_{F_2}$, and $\theta_G$.
1: procedure BATM ($\theta_{F_1}, \theta_{F_2}, \theta_G$)
2:     Initialize the parameters $\theta_{F_1}, \theta_{F_2}$, and $\theta_G$ randomly
3:     for $i := 1$ to $T$ do
4:         /* Step 1 */
5:         Compute the loss $L_{\text{bce}}$
6:         $\theta_{F_1} = \theta_{F_1} - \alpha \frac{\partial L_{\text{bce}}}{\partial \theta_{F_1}}$
7:         $\theta_{F_2} = \theta_{F_2} - \alpha \frac{\partial L_{\text{bce}}}{\partial \theta_{F_2}}$
8:         $\theta_G = \theta_G - \alpha \frac{\partial L_{\text{bce}}}{\partial \theta_G}$
9:         /* Step 2 */
10:          /* Predicting pseudo-labels for target projects */
11:          $\rho = p(y|x_T) > 0.5?1:0$
12:          Compute the loss $L_{s,t}$
13:          $\theta_{F_1} = \theta_{F_1} - \alpha \frac{\partial L_{s,t}}{\partial \theta_{F_1}}$
14:          $\theta_{F_2} = \theta_{F_2} - \alpha \frac{\partial L_{s,t}}{\partial \theta_{F_2}}$
15:          $\theta_G = \theta_G - \alpha \frac{\partial L_{s,t}}{\partial \theta_G}$
16:     end for
17: end procedure
```

Algorithm 1: BATM training algorithm.

achieve a "balance" between classifiers, thereby reducing the uncertainty of prediction.

Additionally, it is important to assess the disparity between the predictions and labels of the main classifier F_1. For this purpose, the binary cross-entropy loss is employed and formalized as follows:

$$L_{ce} = E\left[-\widehat{p}_t^j \log_2 F_1\left(x_t^j|\theta_t\right)\right]. \tag{10}$$

Given that the target project has no actual label, $\widehat{p}_t^j$ is considered a pseudo-label in this context. The paper incorporates a bias regularization term to enhance learning from potentially noisy labels. This regularization term can be formulated as follows:

$$L_{F_1F_2} = E[\exp\{-D_{kl}\}L_{ce} + D_{kl}]. \tag{11}$$

For the average output of classifiers F_1 and F_2, we confuse the classifier by maximizing the entropy of this output. Our expectation is for the classifier to not accurately identify the generated samples and real samples, thus aligning the distribution. We calculate the entropy loss of the classifier output using the following formula:

$$L_H = E[-p(y|x_S)\log_2 p(y|x_S)]. \tag{12}$$

Based on the above adjusted adversarial training [31] generator and classifier, we construct a total loss function to perform backpropagation to update parameters. Specifically, the comprehensive, objective function of our BATM method can be formulated as follows:

$$\begin{aligned} &\min_G \max_{F_1,F_2} L_{\text{all}} \\ &L_{\text{all}} = L_{\text{bce}} + \lambda L_{\text{mdd}} + L_{F_1F_2} - L_H. \end{aligned} \tag{13}$$

In the formula, λ is used to find the optimal parameters through cross-validation. After a certain number of iterations, the multimodal distributions between source and target projects are effectively aligned. Algorithm 1 describes the pseudocode of BATM. We repeat these two steps during the training phase. We use a variance regularization term to correct learning from noisy labels (Step 1), and then adversarially train the classifiers and generator after pseudo-labeling the target projects (Step 2).

4. Experiment Settings

This paper employs the machine learning framework PyTorch for the implementation of the BATM model. Specifically, the number of heads in the multihead attention model of the transformer network is configured to be 16. Throughout the training process, a batch size of 32 is utilized. The training process utilizes a stochastic gradient descent, and the Adam optimizer [32] is employed to update parameters. The default momentum parameter (0.9) in Adam and the initial learning rate is set to 0.001.

Table 1: Eleven projects were picked from the PROMISE dataset.

Project title	Version	# instance	Bug rate (%)
Ant	1.7	745	22.3
Camel	1.6	965	19.5
Forrest	0.8	32	6.3
Ivy	2.0	352	11.4
Log4j	1.2	205	92.2
Lucence	2.4	340	59.7
Poi	3.0	442	63.6
Synapse	1.2	256	33.6
Velocity	1.6.1	229	34.1
Xalan	2.7	909	98.8
Xerces	1.4.4	588	74.3

Table 2: Twenty handcrafted features.

Metric	Description
WMC	This measures the average number of weighted methods in a class [34]
DIT	It represents the number of direct and indirect parent classes [34]
NOC	This feature counts the number of direct subclasses a class has [34]
CBO	It indicates the level of association between a class and other classes [34]
RFC	It represents the total number of methods a class can respond to [34]
LCOM	It indicates the level of relatedness and sharing among methods in a class [34]
LCOM3	An enhanced version of LCOM that calculates the sharing of member variables among methods more accurately [34]
NPM	This counts the total number of public methods in a class [35]
LOC	It measures the total number of lines of code in a class [35]
DAM	It measures the degree to which a class depends on external classes [35]
MOA	It measures the degree to which a class aggregates (contains) other classes [35]
MFA	It measures the proportion of abstract methods in a class [35]
CAM	It measures the level of cohesion among methods in a class [35]
IC	It measures the degree to which a class depends on inheritance [33]
CBM	It measures the level of method invocation between methods in a class [33]
AMC	It represents the average complexity of methods in a class [33]
CA	It represents the number of classes referencing a class [36]
CE	It represents the number of classes referenced by a class [36]
Max CC	It represents the highest complexity of a method within a class [37]
Avg CC	It represents the average complexity of all methods in a class [37]

4.1. Benchmark Datasets. In recent years, researchers [5, 8] have shown a preference for utilizing open-source software projects in CPDP research. To evaluate the BATM model proposed in this paper, 11 Java projects from the PROMISE [33] benchmark dataset were selected. This particular open-source software repository is widely utilized in the CPDP field.

This paper carefully chooses 11 open-source software projects from the PROMISE dataset (refer to Table 1). In this dataset, researchers carefully enumerated and computed 20 handcrafted features (see Table 2), with a predominant focus on program complexity [36]. Within the scope of the CPDP problem, defect prediction necessitates the specification of both a source project and a target project. Thus, this paper initially designates a project as the target. Then, it utilizes the remaining projects in the dataset as sources to establish a CPDP task pair originating from the source project to the target project. Following this approach, 110 sets of CPDP task pairs can be constructed within the PROMISE dataset, forming the basis for the experiments in this paper.

4.2. Evaluation Metrics. To assess the reliability and effectiveness of the BATM model, we employ three widely used evaluation metrics: F_1 measure, balanced accuracy, and area under curve (AUC) [38]. In CPDP, it is customary to utilize the confusion matrix (Table 3) for evaluating the classifier's performance.

The effectiveness of a classifier on positive classes can be assessed using sensitivity (or recall), while its effectiveness on negative classes can be evaluated using specificity. Precision is the metric used to assess the accuracy of a model. Below, we provide the calculation formulas for these indicators:

Table 3: Confusion matrix.

Actuality	Forecast results	
	Classified positive	*Classified negative*
Actual positive	TP (true positive)	FN (false negative)
Actual negative	FP (false positive)	TN (true negative)

$$\begin{aligned} \text{Sensitivity} &= \frac{\text{TP}}{\text{TP}+\text{FN}} \\ \text{Specificity} &= \frac{\text{TN}}{\text{TN}+\text{FP}} \\ \text{Precision} &= \frac{\text{TP}}{\text{TP}+\text{FP}}. \end{aligned} \tag{14}$$

The F_1 measure, being the harmonic mean of sensitivity and precision, offers a balanced evaluation of these two indicators. It aims to find the optimal balance between making accurate positive predictions and capturing all positive instances. The formal description of F_1 measure is as follows:

$$F_1 = \frac{2 \times \text{Sensitivity} \times \text{precision}}{\text{Sensitivity} + \text{precision}} = \frac{2\text{TP}}{2\text{TP}+\text{FN}+\text{FP}}. \tag{15}$$

In CPDP tasks, classifiers that predict all samples as majority classes can perform well in accuracy (ACC). Therefore, this paper employs balanced accuracy (BA) to gauge the validity of the CPDP model. It provides a comprehensive evaluation considering the classifier's performance in both majority and minority classes. The calculation expression for balanced accuracy is as follows:

$$\begin{aligned} \text{Balanced accuracy} &= \frac{\text{Sensitivity} \times \text{specificity}}{2} \\ &= \frac{\text{TP} \times \text{TN}}{2(\text{TP}+\text{FN})(\text{TN}+\text{FP})}. \end{aligned} \tag{16}$$

To comprehensively compare the validity of BATM and the baseline methods, this chapter employs the AUC evaluation index [38]. AUC stands for the area under the curve. It is the area of a 2D graph illustrating the connection between the true positive rate and the false positive rate. The y-axis denotes the true positive rate, while the x-axis represents the false positive rate. The ROC curve is generated by adjusting the classification threshold across all possible value ranges, thereby distinguishing between instances with defects and those without. An effective predictor should yield an AUC value as close to 1 as possible. Additionally, ROC analysis shows strength, particularly in situations with class imbalance and uneven classification costs. Hence, this metric is well-suited for applications in CPDP contexts.

4.3. Baseline Methods. We benchmark the BATM method against the following nine baseline methods:

(i) LR: It relies on handcrafted features as input for the LR classifier in the prediction process.

(ii) NNFilter [39]: Only handcrafted features are utilized, and similar instances are aggregated to construct a training set.

(iii) TCA [14]: It maps data from different projects together into a latent space. By minimizing the distance between them, data migration is accomplished while preserving the geometric structure and data variance intact.

(iv) TCA+ [15]: This is a transferable feature learning method. TCA is expanded by exclusively incorporating handcrafted features along with customized normalization rules.

(v) DBN [6]: This method employs a specific network to handle serialized AST and generate semantic features.

(vi) DPCNN [12]: This method utilizes a CNN to process serialized AST and extract semantic features. It also integrates handcrafted features into the process.

(vii) TCNN [5]: It added a matching layer based on CNN for feature mapping and utilized MMD to mitigate differences between various projects.

(viii) MANN [17]: This method employs serialized AST and handcrafted feature generation vectors. It implements multicore MMD to align the feature distributions across different projects.

(ix) ADA [20]: It is based on adversarial training and uses two classifiers to detect target project samples that are far away from the support vectors of source project samples. This is done to obtain the relationship between these samples and classification boundaries and perform distribution alignment.

This paper utilizes PyTorch to replicate all baseline models, excluding TCNN, as its source code is publicly available. Following that, the paper retrains these baseline models under approximately similar experimental conditions. Taking into account the class imbalance challenge in CPDP [40], ADA actually uses oversampling for both source and target projects. However, we believe that oversampling the target project actually uses the label information leaked by the target project, which is deemed unreasonable. Therefore, we apply the random oversampling method to source projects in all models without any manipulation of the target projects. This is done to ensure the utmost fairness in the comparative experiments.

4.4. Statistical Analysis Methods. To ascertain the significance of the performance comparison results, this chapter employs the Wilcoxon signed-rank test. It is a nonparametric statistical hypothesis test for the evaluation index. In all test cases, the null hypothesis posits that there is no statistical discrepancy in the performance results between the two compared methods. The chosen statistical significance level α is set at 0.05. A p-value below 0.05 indicates a noticeable distinction, while a p-value above 0.05 suggests that the distinction is not statistically significant.

Table 4: The results of the F_1 measure comparison with nine methods.

Target project	LR	NNFilter	TCA	TCA+	DBN	DPCNN	TCNN	MANN	ADA	Ours
Ant	0.445	0.443	0.444	0.418	0.367	0.425	0.410	0.515	0.506	**0.547**
Camel	0.328	0.325	0.326	0.328	0.310	0.333	0.335	**0.405**	0.401	0.398
Forrest	0.192	0.167	0.114	0.120	0.131	0.152	0.133	0.229	0.256	**0.274**
Ivy	0.289	0.274	0.258	0.250	0.229	0.268	0.268	0.254	0.332	**0.455**
Log4j	0.435	0.637	0.661	0.650	0.654	0.659	0.686	0.467	0.594	**0.705**
Lucene	0.603	0.597	0.619	0.551	0.584	**0.641**	0.632	0.605	0.543	0.615
Poi	0.508	0.634	0.605	0.589	0.612	0.622	**0.669**	0.634	0.512	0.506
Synapse	0.525	0.520	0.522	0.549	0.453	0.520	0.495	**0.584**	0.569	0.560
Velocity	0.506	0.501	0.496	0.310	0.453	0.517	0.500	0.515	0.461	**0.518**
Xalan	0.534	0.605	0.647	0.655	0.651	0.625	**0.680**	0.572	0.588	0.622
Xerces	0.532	0.611	0.607	0.577	0.590	0.605	0.620	0.584	0.587	**0.669**
Average	0.445	0.483	0.482	0.455	0.458	0.487	0.494	0.488	0.486	**0.533**
W (T) (L)	10/0/1	10/0/1	8/0/3	9/0/2	9/0/2	8/0/3	8/0/3	8/0/3	8/0/3	—
Improvement	19.8%	10.3%	10.7%	17.3%	16.5%	9.4%	8.0%	9.3%	9.6%	—
p-Value	0.004	0.041	0.042	0.026	0.026	0.045	0.048	0.047	0.016	—

To facilitate pairwise comparisons between methods, this chapter employs a "Win/Tie/Loss" analysis. This analysis of a specific performance metric between our method and the comparison method yields three possible outcomes. These outcomes indicate the number of instances where our method outperforms, performs equivalently to, or underperforms compared to the comparison method, respectively.

Additionally, this paper applies a variant of the Scott-Knott effect size difference (ESD) test [41] to quantify the effect size of the discrepancy between the two methods. The Scott-Knott ESD is evaluated based on this measure. All compared models are arranged in order. The Scott-Knott ESD test is a mean relative analysis that utilizes hierarchical clustering to categorize methods with statistically noticeable distinctions into distinct clusters or groups.

4.5. Research Question. To evaluate the efficacy and performance of the proposed BATM method, we discuss the following two research questions (RQs).

RQ1: Has the BATM method achieved better predictive performance compared to the relevant CPDP methods?

Motivation: Comparison with the CPDP benchmark model is an intuitive way to prove the effectiveness of BATM. We selected nine benchmark models for comparative experiments with BATM.

RQ2: How do BATM components affect its prediction performance?

Motivation: The transfer learning method is an important skill for handling CPDP tasks. We must confirm whether the relevant components in the BATM model can effectively help the model transfer the learned knowledge, thereby improving the predictive performance of the model.

5. Experimental Results

This section carefully designs and executes a substantial number of experiments to validate the accuracy and efficacy of the model put forward in this paper. It also examines the prediction performance of BATM founded on the obtained experimental results. For the experimental results Tables 4–9, each row related to method performance contains a bold value, indicating the best-performing method in the same target project.

5.1. RQ1: Has the BATM Method Achieved Better Predictive Performance Compared to the Relevant CPDP Methods? Tables 4–6 present the comparative experimental outcomes on the PROMISE dataset. It is evident that the BATM model put forward in this paper demonstrates noteworthy performance, achieving average scores of 0.533, 0.714, and 0.698 in the F_1 measure, balanced accuracy, and AUC, respectively. This places it at the forefront among all compared models. In terms of Win/Tie/Lose ($W/T/L$) analysis, BATM outperforms the baseline method in F_1 measure for a minimum of 8 projects, in balanced accuracy for a minimum of 9 projects, and in AUC for a minimum of 10 projects. This demonstrates that our proposed method exhibits superior prediction performance compared to the baseline method across most projects. To provide a clearer illustration of how our method surpasses other baseline methods, we quantify the improvement in model performance.

From the F_1 measure, BATM achieves optimal performance in most of the projects and improves the performance by 8.0%–19.8% on average over the benchmark models. Specifically, BATM improves 14.5% on average over LR, NNFilter, TCA, and TCA+ using only handcrafted features; 16.5% over DBN using only semantic features; 9.4% over DPCNN using joint features, and 8.9% over TCNN, MANN, and ADA using joint features and dealing with the differences in the distributions among data.

In terms of balanced accuracy, BATM achieves superior performance in most projects, with an average performance that is 10.1%–38.3% higher than the baseline model. It is worth noting that the balanced accuracy of the BATM model on projects Forrest and Log4j only reached 0.403 and 0.380, respectively. This article believes that it may be because the

Table 5: The results of the balanced accuracy comparison with nine methods.

Target project	LR	NNFilter	TCA	TCA+	DBN	DPCNN	TCNN	MANN	ADA	Ours
Ant	0.547	0.589	0.590	0.562	0.525	0.558	0.543	0.620	0.772	**0.873**
Camel	0.507	0.531	0.512	0.516	0.509	0.515	0.511	0.606	0.692	**0.739**
Forrest	0.516	0.484	0.454	0.399	0.469	0.485	0.449	**0.561**	0.445	0.403
Ivy	0.511	0.562	0.533	0.544	0.508	0.540	0.537	0.586	0.743	**0.896**
Log4j	0.481	0.486	0.507	0.501	0.503	0.506	**0.536**	0.461	0.440	0.380
Lucene	0.571	0.564	0.575	0.544	0.536	0.591	0.582	0.536	0.682	**0.711**
Poi	0.521	0.595	0.554	0.543	0.555	0.604	0.602	0.600	0.704	**0.745**
Synapse	0.524	0.598	0.584	0.620	0.531	0.574	0.553	0.640	0.753	**0.818**
Velocity	0.523	0.585	0.560	0.548	0.529	0.571	0.549	0.603	0.626	**0.780**
Xalan	0.438	0.439	0.481	0.493	0.488	0.514	0.521	0.504	0.651	**0.796**
Xerces	0.539	0.581	0.563	0.540	0.533	0.581	0.553	0.506	0.634	**0.712**
Average	0.516	0.547	0.538	0.528	0.517	0.549	0.540	0.566	0.649	**0.714**
W (T) (L)	9/0/2	9/0/2	9/0/2	10/0/1	9/0/2	9/0/2	9/0/2	9/0/2	9/0/2	—
Improvement	38.3%	30.6%	32.8%	35.2%	38.1%	30.1%	32.3%	26.2%	10.1%	—
p-Value	0.008	0.008	0.008	0.006	0.008	0.010	0.013	0.013	0.026	—

Table 6: The results of the AUC comparison with nine methods.

Target project	LR	NNFilter	TCA	TCA+	DBN	DPCNN	TCNN	MANN	ADA	Ours
Ant	0.608	0.632	0.635	0.612	0.556	0.616	0.600	0.632	0.742	**0.800**
Camel	0.528	0.545	0.540	0.543	0.524	0.548	0.549	0.616	0.632	**0.678**
Forrest	0.457	**0.627**	0.499	0.540	0.544	0.604	0.552	0.579	0.487	0.480
Ivy	0.619	0.620	0.615	0.611	0.563	0.620	0.621	0.635	0.817	**0.869**
Log4j	0.422	0.493	0.424	0.450	0.475	0.477	0.449	0.512	0.518	**0.526**
Lucene	0.477	0.570	0.574	0.560	0.533	0.587	0.579	0.490	0.631	**0.672**
Poi	0.601	0.599	0.563	0.551	0.553	0.589	0.590	0.616	0.671	**0.705**
Synapse	0.523	0.610	0.606	0.636	0.543	0.602	0.578	0.630	0.702	**0.775**
Velocity	0.546	0.591	0.577	0.496	0.541	0.595	0.576	0.607	0.655	**0.736**
Xalan	0.609	0.669	0.640	0.676	0.654	0.707	0.660	0.624	0.687	**0.743**
Xerces	0.534	0.594	0.573	0.561	0.531	0.576	0.541	0.631	0.634	**0.689**
Average	0.539	0.595	0.568	0.567	0.547	0.593	0.572	0.597	0.652	**0.698**
W (T) (L)	11/0/0	10/0/1	10/0/1	10/0/1	10/0/1	10/0/1	10/0/1	10/0/1	10/0/1	—
Improvement	29.5%	17.3%	22.9%	23.1%	27.6%	17.8%	21.9%	16.8%	7.1%	—
p-Value	0.003	0.026	0.004	0.004	0.006	0.016	0.004	0.013	0.004	—

Table 7: Ablation study on the metric of F_1 measure.

Target project	No-HF	No-MC	No-MDD	BATM
Ant	0.545	0.463	0.474	**0.547**
Camel	0.333	0.324	0.338	**0.398**
Forrest	0.262	0.229	0.265	**0.274**
Ivy	0.438	0.387	0.419	**0.455**
Log4j	0.649	0.493	0.574	**0.705**
Lucene	0.607	0.549	0.582	**0.615**
Poi	0.457	0.435	0.432	**0.506**
Synapse	0.521	0.498	0.511	**0.56**
Velocity	0.457	0.411	0.432	**0.518**
Xalan	0.563	0.552	0.579	**0.622**
Xerces	0.551	0.432	0.481	**0.669**
Average	0.489	0.434	0.462	**0.533**
Improvement	8.99%	22.8%	15.4%	—
p-Value	0.003	0.003	0.003	—

Table 8: Ablation study on the metric of balanced ACC.

Target project	No-HF	No-MC	No-MDD	BATM
Ant	0.864	0.859	0.861	**0.873**
Camel	0.72	0.705	0.716	**0.739**
Forrest	0.308	0.253	0.384	**0.403**
Ivy	0.855	0.828	0.804	**0.896**
Log4j	0.325	0.322	0.347	**0.38**
Lucene	0.701	0.664	0.697	**0.711**
Poi	0.74	0.625	0.655	**0.745**
Synapse	0.814	0.761	0.796	**0.818**
Velocity	0.712	0.679	0.686	**0.78**
Xalan	0.751	0.731	0.745	**0.796**
Xerces	0.658	0.623	0.705	**0.712**
Average	0.677	0.641	0.672	**0.714**
Improvement	5.46%	11.4%	6.25%	—
p-Value	0.003	0.003	0.003	—

Table 9: Ablation study on the metric of AUC.

Target project	No-HF	No-MC	No-MDD	BATM
Ant	0.781	0.736	0.775	**0.801**
Camel	0.653	0.628	0.639	**0.678**
Forrest	0.392	0.283	0.437	**0.48**
Ivy	0.856	0.765	0.817	**0.869**
Log4j	0.516	0.447	0.496	**0.526**
Lucene	0.638	0.616	0.639	**0.672**
Poi	0.676	0.654	0.663	**0.705**
Synapse	0.728	0.706	0.723	**0.775**
Velocity	0.648	0.621	0.635	**0.736**
Xalan	0.699	0.687	0.705	**0.743**
Xerces	0.618	0.568	0.601	**0.689**
Average	0.652	0.611	0.648	**0.698**
Improvement	6.56%	14.2%	7.71%	—
p-Value	0.003	0.003	0.003	—

defect rate of Forrest is only 6.3%, and the defect rate of Log4j is as high as 92.2%. However, they are projects with relatively small sample sizes in the PROMISE data set used, with only 32 and 205 code instances, respectively. The class imbalance phenomenon is extremely serious. Even if random oversampling is used for the source project, it will simply repeat the sample. A minority class that lacks diversity will make BATM fall into the dilemma of not learning generalization knowledge and causing overfitting. However, the defect rate of the Xalan project is as high as 98.8%, with 909 data samples. This allows BATM to learn enough knowledge in the project, resulting in a balanced accuracy of up to 0.796, which is at least 52.8% higher than the baseline model. This demonstrates that BATM can still perform well even with imbalanced data distributions, provided there are sufficient samples.

From an AUC perspective, BATM achieves optimal performance in all projects except for Forrest, with an average performance that is 7.1%–29.5% higher than the baseline model. In comparison with models employing joint features, BATM improves DPCNN, TCNN, MANN, and ADA by 17.8%, 21.9%, 16.8%, and 7.1%, respectively.

From the standpoint of statistical hypothesis testing, the *p*-values derived from the Wilcoxon signed-rank test for the three metric indicators of all comparison models are consistently below 0.05. This indicates that the differences between these models and the BATM method are statistically significant at the 95% confidence level.

To visually illustrate the differences between BATM and the comparative models, this paper employs a box plot based on the Scott-Knott ESD test. The methods are categorized and ranked based on the Scott-Knott ESD test results. In Figure 4, the orange line represents the median of the method, and the green triangle represents the mean. Methods toward the front are considered better, and there is not a significant difference in performance among several methods

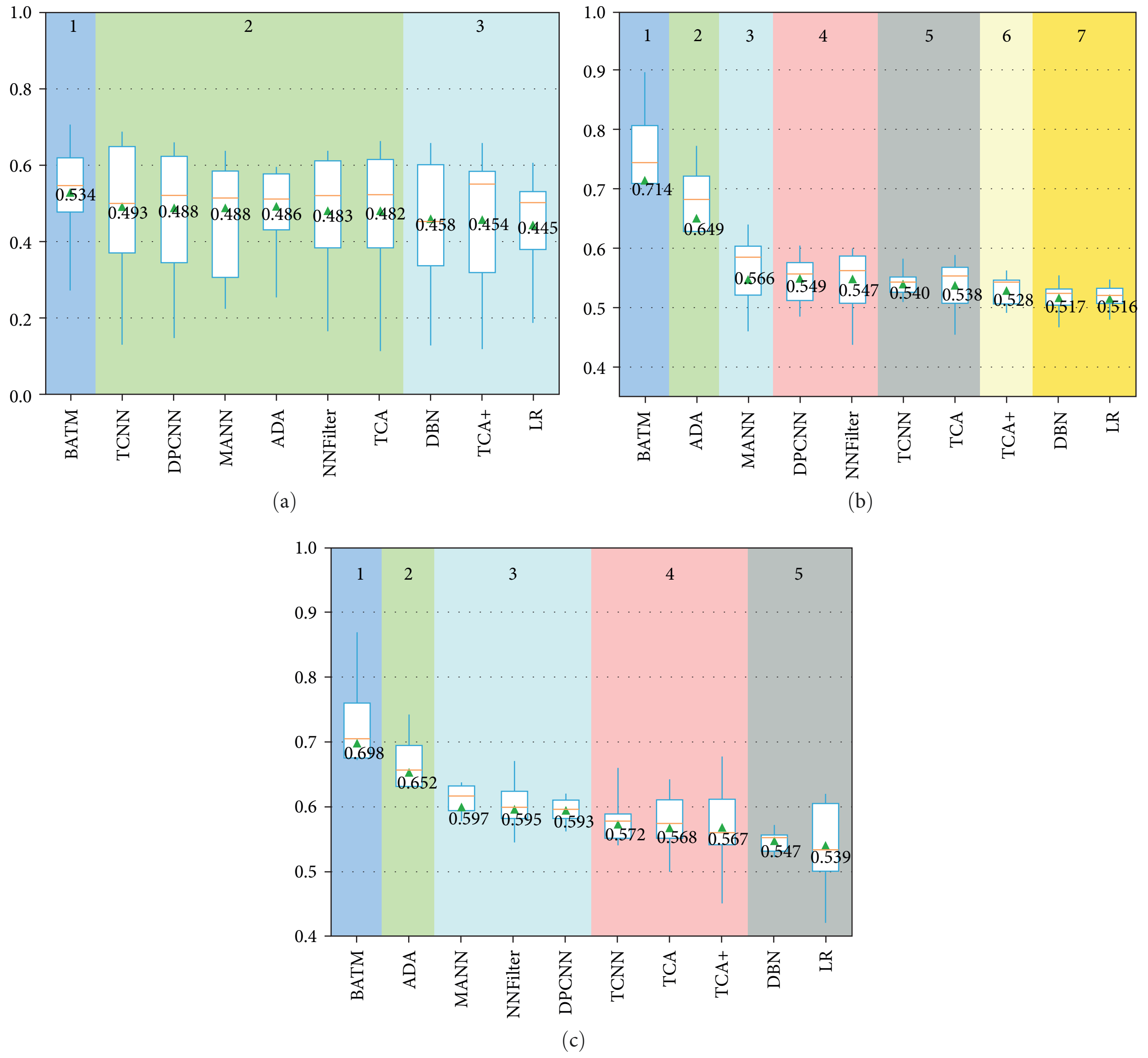


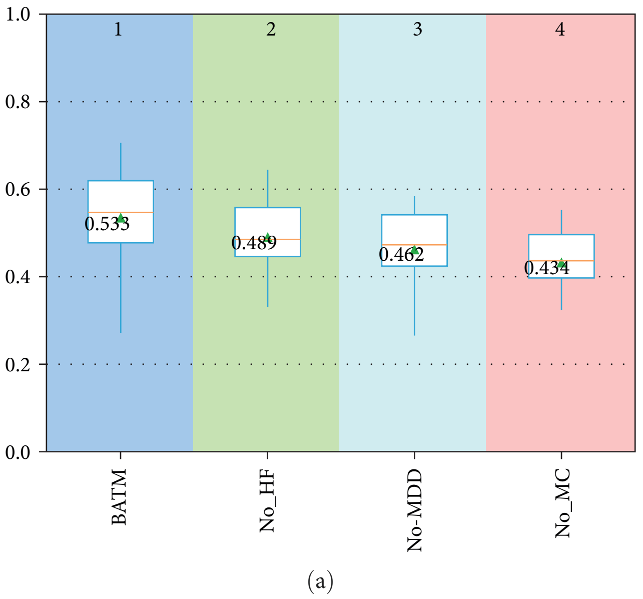
Figure 4: Box plot of three evaluation indicators: (a) F_1 measure on PROMISE dataset; (b) balanced accuracy on PROMISE dataset; (c) AUC on PROMISE dataset.

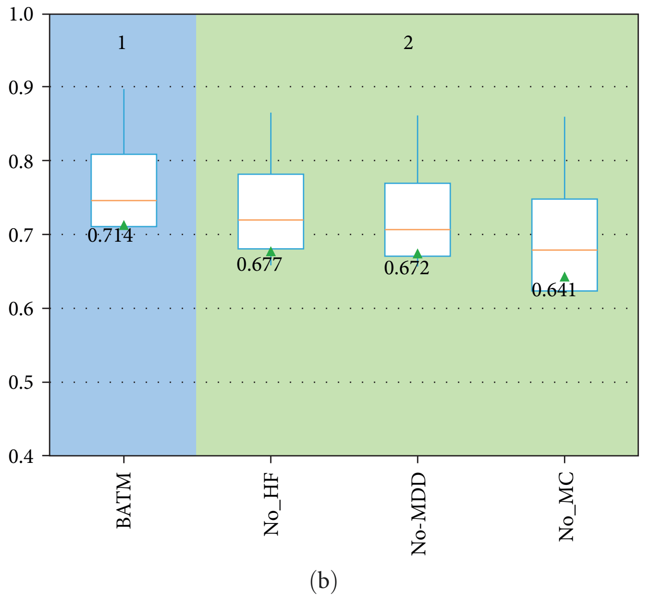
within the same color box. In terms of rankings, BATM secures the top position.

Compared to CPDP methods based on unimodal distribution, BATM excels at capturing the multimodal distribution of code instances. Additionally, employing MDD not only effectively aligns the distribution but also facilitates better differentiation of defective instances. This paper posits that this is a primary factor contributing to BATM's superior predictive performance. Through a detailed and comprehensive observation of the experimental results, we can answer RQ 1: Compared with the related CPDP method, the BATM method achieves better prediction performance.

5.2. RQ2: How Do BATM Components Affect Its Prediction Performance? To gain a deeper understanding of the impact of various components on the model, we performed ablation experiments to assess the performance effects of each component in BATM. These components include handcrafted features (HF), multilinear conditioning (MC), and the MDD method. Specifically, NoHF represents a model that does not utilize handcrafted features for training; No-MC represents a model without multilinear conditioning, and No-MDD represents a model without MDD.

As evident from Tables 7–9, the removal of any of these components results in a significant drop in the defect predictor's performance. In terms of the F_1 measure, BATM attains an average performance of 0.533. In contrast, models without handcrafted features (No-HF), without multilinear conditioning (No-MC), and without maximum density divergence (No-MDD) only achieve average performances of 0.489, 0.434, and 0.462, respectively. This represents performance drops of 8.26%, 18.6%, and 13.3%, respectively.





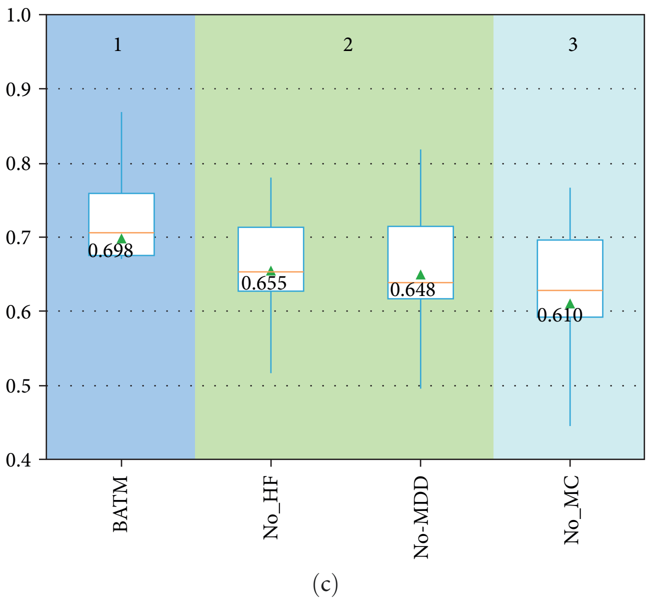


FIGURE 5: Box plot of three evaluation indicators in ablation experiments: (a) F_1 measure; (b) balanced accuracy; (c) AUC.

Concerning balanced accuracy, BATM achieves an average performance of 0.714, while No-HF, No-MC, and No-MDD achieve average performances of 0.677, 0.641, and 0.672, respectively. In terms of AUC measurement, BATM attains an average performance of 0.698, whereas No-HF, NoMC, and No-MDD only reach average performances of 0.652, 0.611, and 0.648, respectively. Figure 5 illustrates the outstanding performance of BATM through a box plot.

The absence of handcrafted features leads to the loss of crucial feature information. This hinders the model's ability to comprehensively learn reliable information, thereby impacting the prediction performance. Without considering the aligned multimodal distribution, regardless of the amount of information the model learns, it may only capture noise. Consequently, the No-MC model experiences the most significant performance decrease compared to other ablation models. The omission of the MDD method means that, even if the model captures the multimodal distribution of the data, it cannot effectively measure distribution differences. Consequently, the model lacks the ability to gradually align these differences, resulting in a significant decline in prediction performance.

Through these experiments, the necessity of combining handcrafted features and employing a multilinear conditioning method based on MDD is verified. Through the above discussion, we are able to answer RQ2: The components of BATM contribute to its good prediction performance.

6. Discussions

In this section, we explore diverse issues related to the BATM approach and discuss potential threats to the validity of this work.

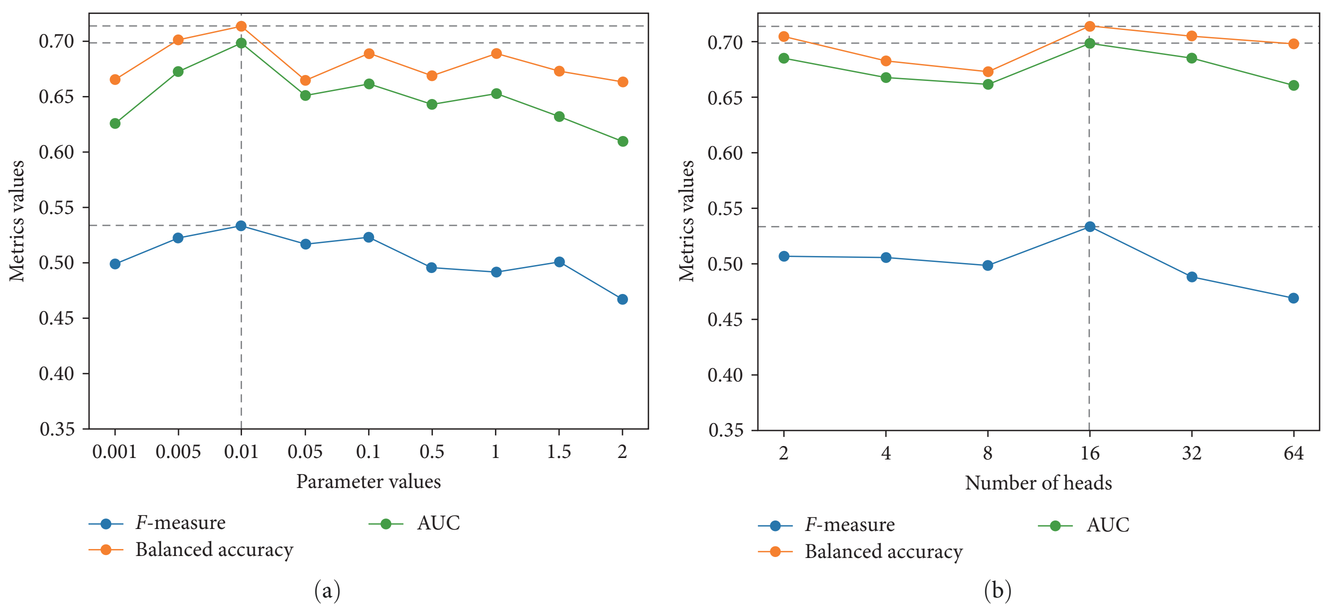


FIGURE 6: Model analysis on PROMISE: (a) the performance of BATM under different parameter values; (b) the performance of BATM under different number of heads.

6.1. How Do the Parameter λ and the Number of Heads of Attention Affect the Performance of the BATM? In this section, we design correlation experiments to explore the effect of the weight parameter λ of MDD and the number of heads of the transformer on BATM.

For the weight parameter λ of the MDD, we vary the value of λ within the range of {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 1.5, 2}. Subsequently, we report the corresponding experimental results. To achieve this, we perform 30 random runs and present the average F_1 measure, balanced accuracy, and AUC for 110 CPDP task pairs. This paper selects the parameter values through a comparison of model performance under different values of the parameter λ. As depicted in Figure 6(a), our proposed BATM method achieves optimal results when λ is set to 0.01. Consequently, we adopt $\lambda = 0.01$ for our experiments. It is noteworthy that 0.01 is a relatively small value. The significance of MDD with minimal weight may be questioned. In light of this concern, we can examine cross-entropy or other entropy-related losses alongside MDD losses. Notably, while entropy involves the logarithm of the output, MDD loss is the sum of squared distances. This leads us to guess that the absolute value of the MDD loss may be considerably larger than the entropy-related loss. Consequently, assigning a smaller weight to the MDD loss guarantees that it is reweighted to the matching magnitude as the entropy-related loss.

For the number of heads of the transformer, we varied it within the range of {2, 4, 8, 16, 32, 64}. Then, we conducted experiments on the PROMISE dataset and reported the experimental results. As displayed in Figure 6(b), the model performance is optimal when the number of attention heads in the transformer is set to 16. If the number of attention heads is too small, each head's capacity to learn information is limited, and it may not fully utilize the input sequence's information. Conversely, if the number of attention heads is too large, especially with a small dataset like PROMISE, it may lead to overfitting the training data and decreased generalization ability. In this study, setting the number of attention heads to 16 allows for both comprehensive learning of the input sequence's information and good generalization ability. This is exactly why we decided to set the number of heads of the transformer to 16.

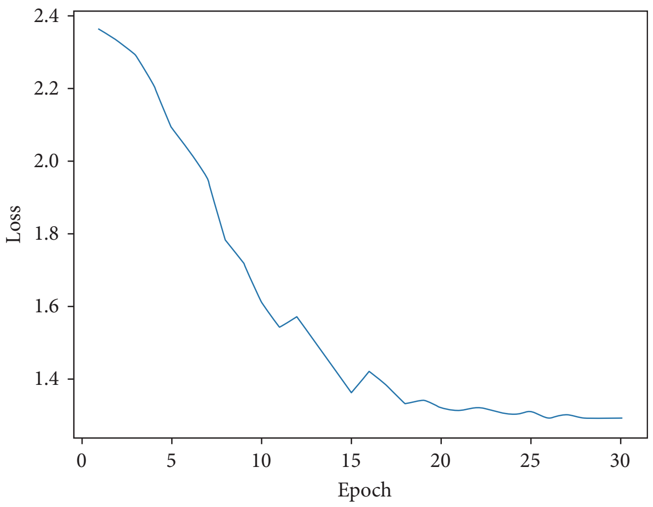


FIGURE 7: The change of loss value.

6.2. Whether the Model Can Learn Useful Knowledge? In deep learning, fluctuations in loss values signify the model's ability to progressively capture essential features and patterns in the data. As illustrated in Figure 7, the line chart depicting the change in loss value reveals a gradual decrease until it plateaus. This observation indicates that our model effectively learns valuable knowledge from the data in a gradual manner until the algorithm reaches convergence. Hence, this article sets the default number of model iterations to 30. This choice strikes a balance between ensuring that the model acquires adequate knowledge and minimizing training time.

6.3. How Does BATM Work by Aligning Multimodal Distributions? Addressing the issue raised concerning features with multimodal distributions, we employ t-SNE for visualizing distribution alignment [42]. t-SNE, a nonlinear dimensionality reduction technique, aids in visualizing the intricate structure of high-dimensional data. By observing the clustering patterns in the t-SNE dimensionality reduction space, we can gain insights into the approximate distribution of the data. In the presence of a multimodal distribution, data points are expected to be distributed across multiple clusters in the reduced dimension space.

As depicted in Figure 8, the sample features of both the source and target projects form two distinct irregular strip clusters, with some less apparent clusters. This suggests that their probability distribution exhibits at least two clear peaks, indicating an overall appearance of a multimodal distribution. Across continuous iterations from the first epoch to the 10th and then to the 25th epoch, the clusters of source and target project features gradually converge and overlap. This reflects the progressive alignment of distributions between them.

By the 25th epoch, the feature clusters of the source and target projects significantly overlap, signifying that their distributions have been effectively aligned. This observation aligns with the conclusion drawn in Figure 7, indicating that the algorithm reaches convergence after the 25th epoch. In summary, BATM demonstrates reliability and effectiveness in aligning the multimodal distributions of features across different projects.

6.4. How Different Classifiers Affect the Performance of BATM? In this subsection, we will discuss why the LR classifier is chosen as the base classifier for BTAM. LR is a commonly used base classifier in CPDP, which is more stable and less likely to overfit data when dealing with unbalanced and small datasets. In order to verify the appropriateness of using LR as a base classifier for BATM, we evaluate the impact of different classifiers on the prediction performance of the model. In addition to LR, we also chose four other classical classifiers, including random forest (RF), K-nearest neighbor (KNN), support vector machine (SVM), and naive Bayes (NB). For RF, KNN, and SVM, we have utilized the common defaults in CPDP, setting the number of decision trees for RF to 100, the value of k for KNN to 5, and using the Gaussian radial basis function as the kernel function for SVM.

In Figure 9, we can observe that the choice of different base classifiers impacts the performance of BATM. When comparing the results of the model on three evaluation metrics using five different base classifiers, we find that LR achieves the best performance in terms of F_1 measure, balanced accuracy, and AUC. SVM comes closest to LR in terms of F_1 measure but significantly lags behind in terms of balanced accuracy and AUC. Additionally, RF and KNN produce comparable results across all three evaluation metrics, while NB is unable to compete with the remaining four base classifiers on any of the three evaluation metrics. To summarize, LR is more suitable as a base classifier for BATM.

6.5. Threats to Validity. While the BATM method proposed in this paper has shown commendable performance in the CPDP task, it still exhibits the following limitations:

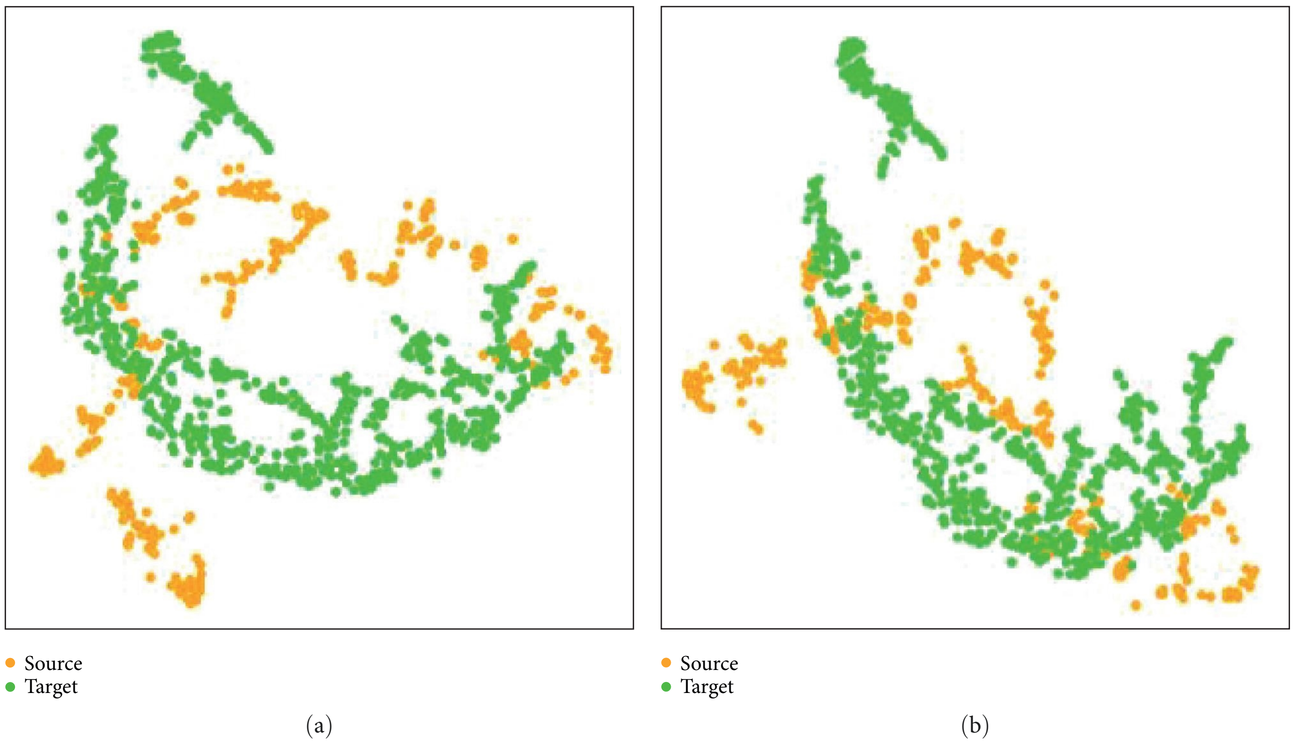
(i) During the experiment, due to the impracticality of evaluating all possible parameter values, our exploration was limited to the impact of some values of the MDD weight parameter on the model performance. However, it is conceivable that there might be more optimal weight parameter values that could result in improved prediction performance.

(ii) For models lacking open-source code, we meticulously implemented them based on the information provided in the respective papers. To ensure a fair comparison, we employed uniform data preprocessing methods and an LR-based classifier. We also took care to select 11 open-source projects from the public benchmark dataset, retraining the models for consistency. However, it is important to note that our implementation may not comprehensively capture all details of the baseline methods.

(iii) To better reveal the multimodal distribution features of the joint features formed by semantic features and handcrafted features, we deliberately chose 11 projects from the PROMISE benchmark dataset for our experiments. However, it is crucial to note that the 11 open-source projects are exclusively coded in Java. Applying our proposed approach to commercial software projects or projects developed in various programing languages may lead to diverse outcomes. The efficacy of our proposed method requires validation on more diverse datasets in future investigations.

7. Conclusions

This paper proposes BATM, a novel CPDP model. BATM effectively addresses the challenge of aligning multimodal distributions between different projects. Initially, the project source codes are parsed into ASTs to obtain token vectors, which are then converted into integer vectors. Next, the transformer network is utilized to acquire semantic features, which are then integrated with handcrafted features to form joint features. Subsequently, the generator and classifier perform adversarial training on the source projects. Afterward, the classifier is used to classify the target projects and assign pseudo-labels. Additionally, the uncertainty of the pseudo-label is adjusted to enhance its accuracy. In short, multilinear conditioning is employed to capture the multimodal distribution of sample features, and MDD is used to measure the distribution differences between projects. This process aligns the distributions and maximizes the density within the same category, ultimately improving the prediction accuracy. Compared to nine other methods, BATM shows improvements of at least 6.6%, 26.2%, and 14.6%



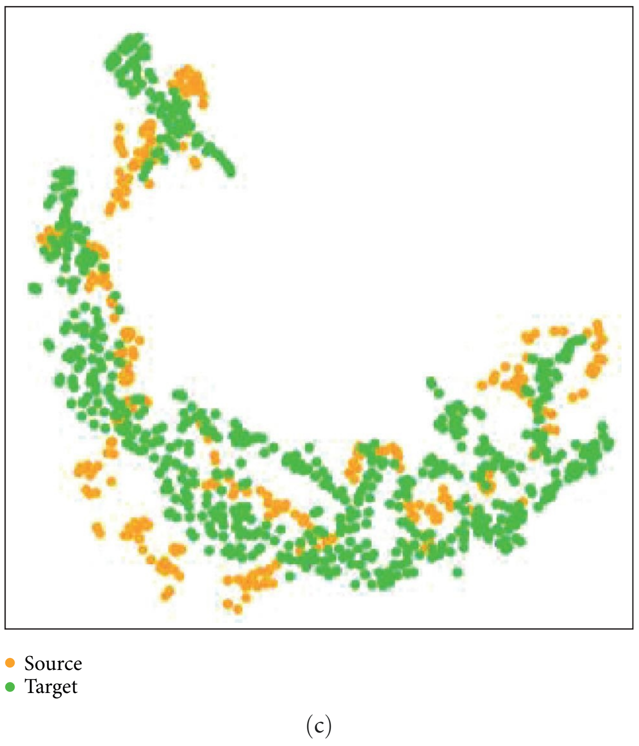


Figure 8: Alignment effect of multimodal distributions between projects at the first iteration (a), the 10th iteration (b), and the 25th iteration (c). Considering the Poi->Camel task pair as an example, with Poi serving as the source project and represented by orange dots, illustrating the sample features within the project. On the other hand, Camel is designated as the target project, and its sample features are depicted by green dots.

in F_1 metric, balanced accuracy, and AUC metric, respectively. BATM also outperforms other methods on most target projects, indicating its exceptional prediction accuracy and generalization capability.

While the method we propose has shown effective performance in the CPDP task, there is still potential for enhancement. Moving forward, we intend to augment the size and diversity of the dataset to mitigate the influence of

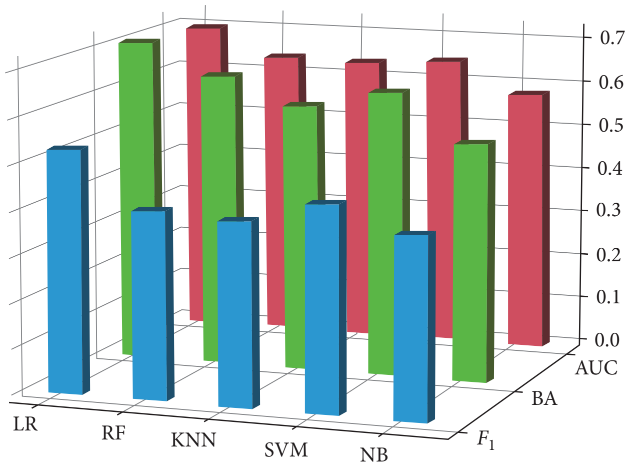


Figure 9: Experimental results of BATM using different base classifiers on PROMISE for F_1 measure (F1); balanced accuracy (BA) and AUC.

the model performance. Additionally, we plan to implement more suitable sampling techniques and data augmentation strategies, especially for source projects with class imbalance or small sample sizes. This is to further improve the model's predictive accuracy.

Data Availability

The PROMISE dataset we use can be accessed at https://doi.org/10.1145/1868328.1868342. The implementation of the proposed *BATM* model is available from the corresponding author upon a reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest in this work.

References

[1] Z. Li, X.-Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," *IET Software*, vol. 12, no. 3, pp. 161–175, 2018.

[2] L. C. Briand, W. L. Melo, and J. Wust, "Assessing the applicability of fault-proneness models across object-oriented software projects," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 706–720, 2002.

[3] 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 *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, pp. 91–100, Association for Computing Machinery, Amsterdam, The Netherlands, 2009, August.

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

[5] S. Qiu, H. Xu, J. Deng, S. Jiang, and L. Lu, "Transfer convolutional neural network for cross-project defect prediction," *Applied Sciences*, vol. 9, no. 13, Article ID 2660, 2019.

[6] S. Wang, T. Liu, J. Nam, and L. Tan, "Deep semantic feature learning for software defect prediction," *IEEE Transactions on Software Engineering*, vol. 46, no. 12, pp. 1267–1293, 2020.

[7] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[8] Y. Xing, X. Qian, Y. Guan, B. Yang, and Y. Zhang, "Cross-project defect prediction based on G-LSTM model," *Pattern Recognition Letters*, vol. 160, pp. 50–57, 2022.

[9] Z. Zhang, C. Jiang, X. Han, and X. X. Ruan, "A high-precision probabilistic uncertainty propagation method for problems involving multimodal distributions," *Mechanical Systems and Signal Processing*, vol. 126, pp. 21–41, 2019.

[10] A. T. Nguyen and T. N. Nguyen, "Graph-based statistical language model for code," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, pp. 858–868, IEEE, Florence, Italy, 2015, May.

[11] C. Liu, W. Sun, W. Chao, and W. Che, "Convolution neural network for relation extraction," in *International Conference on Advanced Data Mining and Applications*, pp. 231–242, Springer, Berlin, Heidelberg, 2013, December.

[12] J. Li, P. He, J. Zhu, and M. R. Lyu, "Software defect prediction via convolutional neural network," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 318–328, IEEE, Prague, Czech Republic, 2017, July.

[13] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Information and Software Technology*, vol. 54, no. 3, pp. 248–256, 2012.

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

[15] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *2013 35th International Conference on Software Engineering (ICSE)*, pp. 382–391, IEEE, San Francisco, CA, USA, 2013, May.

[16] Q. Zou, L. Lu, Z. Yang, X. Gu, and S. Qiu, "Joint feature representation learning and progressive distribution matching for cross-project defect prediction," *Information and Software Technology*, vol. 137, Article ID 106588, 2021.

[17] Q. Huang, L. Ma, S. Jiang et al., "A cross-project defect prediction method based on multi-adaptation and nuclear norm," *IET Software*, vol. 16, no. 2, pp. 200–213, 2022.

[18] S. Tang, S. Huang, C. Zheng, E. Liu, C. Zong, and Y. Ding, "A novel cross-project software defect prediction algorithm based on transfer learning," *Tsinghua Science and Technology*, vol. 27, no. 1, pp. 41–57, 2022.

[19] T. Cheng, K. Zhao, S. Sun, M. Mateen, and J. Wen, "Effort-aware cross-project just-in-time defect prediction framework for mobile apps," *Frontiers of Computer Science*, vol. 16, Article ID 166207, 2022.

[20] H. Song, G. Wu, L. Ma, Y. Pan, Q. Huang, and S. Jiang, "Adversarial domain adaptation for cross-project defect prediction," *Empirical Software Engineering*, vol. 28, Article ID 127, 2023.

[21] J. Deng, L. Lu, and S. Qiu, "Software defect prediction via LSTM," *IET Software*, vol. 14, no. 4, pp. 443–450, 2020.

[22] J. Huang, X. Guan, and S. Li, "Software defect prediction model based on attention mechanism," in *2021 International Conference on Computer Engineering and Application (ICCEA)*, pp. 338–345, IEEE, Kunming, China, 2021, June.

[23] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[24] L. Song, J. Huang, A. Smola, and K. Fukumizu, "Hilbert space embeddings of conditional distributions with applications to dynamical systems," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 961–968, Association for Computing Machinery, Montreal, Quebec, Canada, 2009, June.

[25] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems*, vol. 20, Curran Associates, Inc., 2007.

[26] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 1647–1657, Curran Associates Inc., 2018, December.

[27] P. Kar and H. Karnick, "Random feature maps for dot product kernels," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, pp. 583–591, PMLR, 2012, March.

[28] J. Li, E. Chen, Z. Ding, L. Zhu, K. Lu, and H. T. Shen, "Maximum density divergence for domain adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3918–3930, 2021.

[29] Y. Ding, L. Sheng, J. Liang, A. Zheng, and R. He, "ProxyMix: proxy-based mixup training with label refinery for source-free domain adaptation," *Neural Networks*, vol. 167, pp. 92–103, 2023.

[30] Z. Zheng and Y. Yang, "Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation," *International Journal of Computer Vision*, vol. 129, no. 4, pp. 1106–1120, 2021.

[31] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc., 2014.

[32] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014.

[33] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, pp. 1–10, Association for Computing Machinery, 2010, September.

[34] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[35] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. 4–17, 2002.

[36] R. Martin, "OO design quality metrics," *An Analysis of Dependencies*, vol. 12, no. 1, pp. 151–170, 1994.

[37] T. J. McCabe, "A complexity measure," *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308–320, 1976.

[38] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[39] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, pp. 540–578, 2009.

[40] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[41] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 1–18, 2017.

[42] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.