Journal of Theoretical and Applied Information Technology
15st June 2026. Vol.104. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FEDHCPDP: A FEDERATED SELF-OPTIMIZED MULTI-SCALE RESIDUAL ATTENTION GRAPH NEURAL NETWORK FOR HETEROGENEOUS CROSS-PROJECT DEFECT PREDICTION

[1] RADHALAKSHMI RAJAGOPALAN, [2*] DR. V. RADHA

[1] Research scholar, Department of computer science, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, Tamil Nadu, India, Pin code: 641043

Co Author Mail ID: radha.apac@gmail.co

[2*] Dean, Department of computer science, Avinashilingam Institute for Home Science & Higher Education for Women Coimbatore, Tamil Nadu, India, Pin code: 641043

[2*] Corresponding Author Email Id: radha_cs@avinuty.ac.in

ABSTRACT

Heterogeneous Cross-Project Defect Prediction (H-CPDP) plays a crucial role in software quality assurance by predicting defects in projects with limited historical defect information using knowledge transferred from external source projects. However, existing H-CPDP approaches often suffer from feature inconsistency, distribution mismatch, class imbalance, negative transfer, and privacy concerns. To address these limitations, this study proposes a novel Federated Heterogeneous Cross-Project Defect Prediction (FedHCPDP) framework integrating ranking-guided project selection, federated learning, graph neural networks, residual attention learning, and hybrid bio-inspired optimization. The source projects from dataset are evaluated relative to the target project using Cosine Similarity, Entropy Difference, and Wasserstein Distance to identify the Top-K most relevant projects and reduce negative transfer. Furthermore, the study proposed Self-Optimized Multi-Scale Residual Attention Graph Neural Network (SOMRAGNN) to capture both local and global structural dependencies among software modules through multi-scale graph convolution and residual attention propagation. To further improve adaptive learning capability, the attention coefficients and learning parameters of SOMRAGNN are optimized using a Hybrid Secretary Bird–Bobcat Optimization Algorithm (HSBOA). The locally trained models are aggregated using the proposed Federated Ranking-Guided Weighted Averaging (FedRWAvg) mechanism, where highly relevant projects contribute more significantly to the global model. Extensive experiments conducted on multiple heterogeneous datasets demonstrate that the proposed FedHCPDP framework consistently outperforms baseline models demonstrate the effectiveness, and generalization capability of the proposed framework for reliable heterogeneous software defect prediction.

Keywords: *Heterogeneous Cross-Project Defect Prediction, Federated Learning, Deep Learning, Federated Heterogeneous Cross-Project Defect Prediction, Graph Neural Network, Secretary Bobcat optimization Algorithm.*

1. INTRODUCTION

The demand for software systems that are high-quality, dependable, and maintainable has grown in importance in the ever-evolving software development landscape [1]. Increase in complexity and scale of software systems is necessitating proactive detection of possible defects prior to deployment of systems [2]. The contribution of Software Defect Prediction (SDP) is that it can automatically find the defect-prone modules using the historical software metrics and defect-data to achieve this goal [3] [4] [5]. Proper defect prediction allows developers and testers to spend their resources in the most effective manner, focus their efforts on testing and improve the quality of their software, which lowers the maintenance costs and shortens the time of delivery.

Traditional Within-Project Defect Prediction (WPDP) methods use the access to ample labeled data in the same project to train machine-learning or deep-learning models [6] [7]. Such

models presuppose that the past pattern of defects of a project can be transferred to future versions. But in most of the software developments, especially new or small-scale developments, the labeled defect data are usually limited or absent. This absence of defect information is a limitation to the usefulness of the WPDP models and it is difficult to learn accurate prediction models with no defect information [8] [9]. In order to overcome this limitation, researchers have considered Cross-Project Defect Prediction (CPDP) that uses defect information in source projects to estimate defects in a target project that does not have labeled information [10] [11]. The hypothesis used by CPDP is that software systems frequently have similar structural, functional, or behavioral properties, particularly when they are written with use of similar programming paradigms or frameworks or development environments [12]. Although CPDP extends the predictability range of defects, it creates major obstacles as a result of project heterogeneity different feature spaces, coding styles, development styles, and distribution of data. These differences frequently cause a phenomenon referred to as domain shift, in which the patterns learned in the source projects cannot be applied in the target project, causing poor predictive accuracy.

The current models of CPDP are trying to address these discrepancies using the methods of feature alignment, transfer learning, instance selection and representation learning [13] [14]. Even though these techniques have demonstrated a bit of success, they continue to have challenges with the detection of the complex interdependencies between software modules and the hierarchical relationship between source and target projects [15]. Further, traditional deep learning models consider input features as independent vectors without considering the structural or relational information present in software systems [16]. It is a simplification that causes partial representation learning and non-optimal prediction accuracy. Graph Neural Networks (GNNs) represent an effective paradigm of learning on graph-structured data in recent years [17]. When the software defect prediction is concerned, a module can be modeled as a graph node and the relationships as dependencies, inheritance or function calls can be represented as edges [18]. GNNs can be effective in learning node embeddings by means of message passing which ensures that both local and global structural patterns are captured [19]. Nonetheless, conventional GNNs are frequently weak in multi-scale relationship modelling and importance hierarchy, especially when the knowledge is transferred through software projects of different types.

Despite this, heterogeneous CPDP has been shown to have several challenges due to the difference in feature spaces, data distributions, and coding styles, and development environments across projects [20]. Such inconsistencies tend to result in ineffective knowledge transfer and ineffective model generalization. Current CPDP methods to use instance selection, feature alignment or deep learning models continue to have difficulties in capturing the complex relationship between projects and meeting the domain drift between heterogeneous projects [21]. In addition, most current federated learning-based models applied to cross-project systems are based on basic model averaging models such as FedAvg, and do not account for project relevance, model reliability, and variations in data quality [22]. Consequently, the global aggregation tends to bend towards the major sources and to decrease the predictive accuracy and impartiality of the system. In order to overcome these constraints, this study proposes Federated Heterogeneous Cross-Project Defect Prediction (FedHCPDP) framework that integrates SOMRAGNN model. The key contributions of this study are summarized as the following:

- The study proposes a novel FedHCPDP framework that integrates a ranking-based source project selection mechanism with SOMRAGNN model, enabling adaptive knowledge transfer, enhanced structural dependency learning, and improved generalization across heterogeneous software projects while preserving data privacy.
- The study introduces Self-Optimized Multi-Scale Residual Attention Graph Neural Network (SOMRAGNN) to learn hierarchical structural dependencies and both local and global feature interactions in heterogeneous projects.
- The Hybrid Secretary Bobcat Optimization Algorithm (HSBOA) is introduced to optimize hyperparameters and feature weights in an adaptive manner and enhance the robustness of the model and its convergence.
- The study proposes a Federated Ranking-Guided Weighted Averaging (FedRWAvg) scheme to rank and weights source projects according to their relevance, ensuring meaningful and fairness-oriented aggregation.

The rest of this paper is structured as follows: summary of previous research on CPDP is given in Section 2. The FedHCPDP framework's comprehensive architecture is presented in Section 3. The experimental setup, datasets, evaluation metrics, and comparative performance analysis of the suggested framework versus the most advanced federated and CPDP models are presented in Section 4. The main contributions and conclusions of this study are outlined in Section 5.

2. RELATED WORK

To provide a systematic understanding of current developments, the related work is categorized into four major groups: instance- and feature-based approaches, feature alignment and transfer learning methods, deep learning-based CPDP approaches, and federated learning-based CPDP approaches.

2.1. Instance-based approaches

The initial research on CPDP concentrated on feature engineering and instance selection methods for minimizing the distribution difference between source and target projects. Bal and Kumar [23] developed a unique framework for pre-processing which they termed Unique Selection of Matched Metrics (USMM) and employed the Hungarian algorithm and Kolmogorov–Smirnov (KS) statistical test to find matching metric pairs across projects. USMM increased the effectiveness of the metric matching, but the metric matching framework was mainly pairwise feature based and neglected higher-order structural similarities or distributional heterogeneity between projects. Similarly, Zhang et al. [24] proposed an efficient CPDP framework named IAPCP which involves two phases of correlation alignment and intra-domain programming. The final defect predictor was created using the probabilistic annotation matrix optimization method and covariance minimization technique between feature spaces of source and target. While the technique minimized discrepancies in feature distribution, it made it hard to apply it in software repositories where source-target data sharing is not allowed due to privacy concerns. Abdu et al. [25] introduced hybrid reduction-based CPDP scheme called RH-CPDP. The framework combined the hybrid deep neural networks with Principal Component Analysis (PCA) to deal with multicollinearity and eliminate redundant software metrics. Although RH-CPDP achieved greater prediction stability, it was not flexible enough to be implemented to various heterogeneous projects with different structural properties.

These techniques show that feature engineering and instance selection can slightly increase the effectiveness of transfer but normally use shallow statistical alignment methods and typically do not account for complex structural relationships between software modules.

2.2. Transfer learning-based approaches

A few studies attempted to tackle the distribution mismatch issue better by investigating techniques of feature alignment and feature transfer for CPDP. Zou and Wang [26] introduced a three-stage architecture called TriStage-CPDP, which combines the improvements from GraphSAGE with CodeT5+, Recursive Feature Elimination (RFE), Least Absolute Shrinkage and Selection Operator (LASSO), and Locality Preserving Projection (LPP). TriStage-CPDP enhanced the quality of the representation, but its transfer features were still reliant on centralized learning, and lacked adaptive relevance-based sources selection. To address this issue, Tao et al. [27] proposed a Transfer Long Short-Term Memory (TLSTM) network to incorporate transfer learning into LSTM networks for CPDP. The approach extracted semantic program features using LSTM while Transfer Component Analysis (TCA) aligned transferable metric representations between projects. Finally, Logistic Regression (LR) was used to classify the combined features. The sequential learning structure was not able to well capture the dependency information of software in a graph structure while TLSTM had better capability of semantic feature transfer. To enhance the generalization capability in CPDP, Setiawan and Subekti [28] proposed a Multi-View Neural Network (MVNN) approach that included feature transformation and missing-value imputation with heterogeneous software metrics. Many preprocessing operations were included in the multi-view learning framework, and there was no adaptive graph-based representation learning.

The use of transfer learning and feature alignment techniques that enhance the consistency of cross-project representations, but many of them are restricted in their ability to deal with the case of highly heterogeneous software ecosystems with complex structural dependencies and dynamic feature distribution.

2.3. Deep learning and graph-based approaches

In recent years, deep learning techniques have been increasingly applied to boost the performance of software defect prediction. Javed et al. [29]

presented Smote correlation and attention gated recurrent unit optimized LSTM network (SCAG-LSTM) model. The model was designed to address the issues of class imbalance and sequential feature learning by incorporating the Edited Nearest Neighbors (ENN), Synthetic Minority Oversampling Technique (SMOTE), Correlation-based Feature Selection (CFS), Bi-GRU and Bi-LSTM. While SCAG-LSTM showed the ability to learn the context, it lacked an explicit dependency graph model of software in its sequential architecture. Wang et al. [30] proposed an Improved Clustering with Graph-Embedding Features (IC-GraF) framework for software defect prediction. The approach generated graph-level software representations with an Improved Deep Graph Infomax (IDGI) model and built Program Dependency Graphs (PDGs). The software structure was well represented by the graph embedding techniques but the framework was mainly developed for centralized defect prediction and not a federated and heterogeneous environment. Saraireh et al. [31] presented a Binary White Shark Optimizer (WSO) for enhancing the software defect prediction by ensemble learning. Optimizer applied bio-inspired search mechanisms for improving feature selection and prediction performance. But the framework didn't cover representation learning with a graph or heterogeneous source relevance and privacy preservation.

Despite the contribution of deep learning and graph-based methods to defect representation learning, many current models still have problems of over-smoothing in deep graph-based architectures, poor adaptive optimization, and poor handling of heterogeneous project distributions.

2.4. Federated learning-based approach

Recently, FL has been proposed as a potential approach to privacy-preserving software defect prediction. Inspired by the idea of distributed learning, Chen et al. [32] proposed an Efficient Communication-based Federated Meta-Learning (ECFML) framework, which combined Model Agnostic Meta-Learning (MAML) and lightweight MobileViT networks. While it was shown that the federated CPDP approach is feasible, the aggregation strategy was mainly dependent on the uniform participation of clients and lacked consideration of the relevance and quality of the source projects when aggregating them.

Standard aggregation schemes like FedAvg used in existing federated learning-based CPDP systems mainly rely on dataset sizes to define the aggregation. In other CPDP systems (heterogeneous), however, larger volumes of data do not always apply directly to the specific project. Thus, unrelated or loosely connected source projects may cause bias and adversely affect the generalization capabilities of the global model, which is used to update the global parameters. Furthermore, existing federated CPDP systems seldom investigate the incorporation of structural graph learning, adaptive optimization, and relevance-aware source project selection in a single framework. Most of the current approaches also do not provide any means of dynamic balancing between the local and the global knowledge transfer when the heterogeneity of the distribution is severe.

2.5. Problem statement

Although there have been advances in CPDP, current methods still have a number of significant problems. Most transfer learning approaches and feature alignment methods rely on the assumption of homogeneous project attributes and find it difficult to transfer well to highly heterogeneous repositories. Likewise, most of the deep learning based CPDP models emphasize on sequential or statistical feature learning and neglect to consider graph-based software dependencies like inheritance, coupling and invocation relationships between software modules. Over-smoothing and unstable learning in deeper architectures are also issues with existing graph-based approaches. In addition, there have been limited approaches to privacy-preserving federated learning methods for CPDP because they rely on static aggregation strategies like FedAvg, which ignore whether a particular client source project is relevant or not. This can therefore negatively affect the global model and result in negative transfer if projects are irrelevant or somewhat related. Current hyperparameter tuning techniques are also primarily static and do not adapt the learning behavior flexibly to the varying characteristics of the diverse projects. These restrictions are all detrimental in terms of robustness, scalability and generalization in real world heterogeneous CPDP environments.

Hence, there exists a critical need for an intelligent, privacy-preserving, structurally aware and relevance-guided federated framework that can be used for effectively selecting relevant source projects, modeling software dependency structure, adaptively optimizing learning parameters and improving knowledge transfer across the heterogeneous software repositories. Based on these

challenges, the present study addresses the following research questions:

- RQ1: How can heterogeneous source projects be effectively ranked and selected to minimize negative transfer in cross-project defect prediction?
- RQ2: How effectively can graph neural networks capture multi-scale structural dependencies among software modules for improved defect prediction in heterogeneous environments?
- RQ3: Does a relevance-aware federated aggregation strategy (FedRWAvg) outperform conventional methods such as FedAvg in heterogeneous cross-project defect prediction settings?
- RQ4: Can the proposed hybrid HSBOA optimization method improve model adaptability, convergence stability, and predictive performance across diverse software projects?

3. PROPOSED METHODOLOGY

The proposed FedHCPDP framework designed to address key challenges in heterogeneous cross-project defect prediction, including feature inconsistency, distribution mismatch, class imbalance, negative transfer, and privacy preservation. Figure 1 illustrates the overview of proposed framework. Existing CPDP methods often assume homogeneous feature spaces and centralized data availability, limiting their effectiveness in real-world heterogeneous software environments. To overcome these limitations, the proposed framework integrates ranking-based source project selection, federated learning, graph neural network modeling, residual attention learning, and hybrid bio-inspired optimization into a unified architecture. Initially, source projects from datasets are evaluated relative to the target project using Cosine Similarity, Entropy Difference, and Wasserstein Distance. Based on the obtained relevance scores, the Top-K most relevant projects are selected to minimize negative transfer. To effectively capture structural dependencies among software modules, this study introduces SOMRAGNN. The proposed SOMRAGNN models software projects as dependency graphs and employs multi-scale graph convolution to learn both local and global structural information. In addition, a Residual Attention Mechanism highlights defect-relevant nodes while mitigating over-smoothing in deep graph networks. Furthermore, the attention parameters and learning coefficients of SOMRAGNN are adaptively optimized using HSBOA, which balances exploration and exploitation for effective parameter tuning. Each selected source project independently trains a local SOMRAGNN model, preserving data privacy. The locally trained models are then aggregated using FedRWAvg strategy, where more relevant source projects contribute more significantly to the global model.

Finally, the aggregated global model is deployed on the target project for defect prediction. The overall framework provides a privacy-preserving, structurally aware, and heterogeneity-adaptive solution for reliable cross-project defect prediction.

3.1. Preprocessing

In HCPDP, raw data of several software projects present heterogeneous metrics, features overlap, missing values, and imbalance in the classes. Valuable model training, similarity computation, and federated aggregation cannot take place without appropriate preprocessing. The feature alignment, missing value imputation, normalization, and class imbalance processing are included in the preprocessing pipeline of FedHCPDP. The details of each preprocessing method are described in subsequent section.

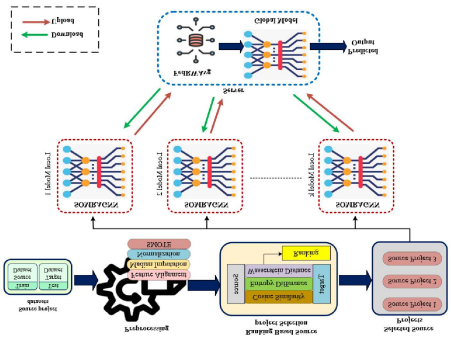

Figure 1: Overview of FedHCPDP Framework

3.1.1. Feature alignment

Since the source and target project differ in their nature, the raw metrics cannot be directly aligned. Assume Z_S and Z_T are the matrix of features of source and target project respectively. Rather than learning non-overlapping features, local projection $f_\emptyset$ is learned on project-specific embeddings as shown in Eqn. (1).

$$E_S = f_\emptyset(Z_S) \qquad E_T = f_\emptyset(Z_T) \tag{1}$$

Where E_s and E_T are project-specific latent embeddings of source and target projects respectively, and $f_\emptyset(\cdot)$ denotes Learnable projection function that maps heterogeneous feature spaces of source or target projects into a common latent space. This is to enable any project to retain all its measurements and a compatibility with the SOMRAGNN architecture and make the similarity of projects based on ranking in a latent space.

3.1.2. Missing Value Imputation

It is expected that software defect datasets may be missing metric values either because the measurement was not completed or because the tools have certain limitations. In the case source project, let $Z_S = [z_{m,j}]$ where $z_{v,j}$ denotes j^{th} metric of module m. The approaching values that are missing are filled in with the median of each column of the features using Eqn. (2).

$$z_{m,j}{}' = \begin{cases} z_{m,j}; & if\ z_{m,j}\ is\ observed \\ Median(z_{m,j}|z_{n,j}); & otherwise \end{cases} \quad (2)$$

Where $z_{n,j}$ denotes value of j^{th} metric for instance u that is used in calculating the median for imputation.

3.2. Normalization

The feature scaling is carried out to ensure that the various ranges and units of the measure of the different projects are factored after median imputation. For each feature value $z_{m,j}{}'$, the normalized feature representation is calculated as Eqn. (3).

$$z_{m,j}{}'' = \frac{z_{m,j}{}' - \mu_j}{\sigma_j} \quad (3)$$

Where $z_{m,j}{}''$ signifies the normalized value of j^{th} feature for m^{th} software module,

μ_j and σ_j are mean and standard deviation of the feature j the given project. This ensures that the similarity in the distance-based measures and GNN attention mechanisms work with similar scales.

3.2.1. SMOTE

The problem with defect datasets is the severe imbalance in classes where defective modules are underrepresented. In order to mitigate this, Synthetic Minority Oversampling Technique (SMOTE) is used on the projects separately. Given a sample of the minority class z_i, SMOTE produces synthetic samples as Eqn. (4).

$$z_{new} = z_i + \delta \cdot (z_{nn} - z_i) \quad (4)$$

Where δ indicates random number sampled from a uniform distribution between 0 and 1, z_{nn} denotes randomly selected nearest neighbor of z_i in feature space. The process is applied until the minority class reaches a desired ratio, ensuring a more balanced dataset for training model.

3.2.2. Source Project Selection

H-CPDP is susceptible to failure when the source projects have varied feature spaces, class distributions and data distributions to that of the target project. Negative transfer can occur through training irrelevant or mismatched source projects, which decreases prediction accuracy. To counter this, FedHCPDP uses a project selection mechanism in form of ranking based on which the relevance of each source project to the target project is determined quantitatively. It consists of three complementary measures which are feature Similarity (Cosine Similarity), Class Balance Similarity (Entropy Difference), Distributional Similarity (Wasserstein Distance). Lastly, these metrics are united together into a single relevance score and the projects that have been ranked highest are picked to be federated trained.

Let $S = \{S_1, S_2, \dots, S_m\}$ set of m source projects, T be target project, Z_S and Z_T are the feature matrix of source and target project respectively, L_S and L_T are class labels (1 means defective and 0 indicates non-defective).

3.2.3. Cosine Similarity

The feature similarity which determines that the structural and the static figures of the source project are in agreement with the target project. $\bar{Z}_S$ and $\bar{Z}_T$ represent mean-normalized feature vectors using Eqn. (5).

$$\bar{Z}_S = \frac{1}{nS}\sum_{j=1}^{nS} Z_{S,j}\,; \qquad \bar{Z}_T = \frac{1}{nT}\sum_{j=1}^{nT} Z_{T,j} \quad (5)$$

Where nS and nT are denotes the number of modules in the source and target projects, respectively, $Z_{S,j}$ and $Z_{T,j}$ are indicates the feature

vector of j^{th} module in the source and target project respectively.

The cosine similarity (CS) between T and S_i is then calculated using Eqn. (6).

$$CS(T, S_i) = \frac{\bar{Z}_S \cdot \bar{Z}_T}{\|\bar{Z}_S\|_2 \cdot \|\bar{Z}_T\|_2} \quad (6)$$

Where $CS(T, S_i)$ signifies the cosine similarity between source project S_i and target project T, and $\|\cdot\|_2$ represents Euclidean norm. The higher values suggest a superior match between features.

3.2.4. Entropy Difference

The issue of imbalance in the classes is a major problem in software defect datasets. Assume, ρ_1^i and ρ_0^i are the fraction of defective and non-defective modules in S_i. Similarly, ρ_1^T and ρ_0^T are the fraction of defective and non-defective modules in the target project. Entropy of individual projects is defined as Eqn. (7).

$$H(S_i) = -\sum_{k\in\{0,1\}} \rho_k^i \log_2 \rho_k^i ; \quad H(T) = -\sum_{k\in\{0,1\}} \rho_k^T \log_2 \rho_k^T \quad (7)$$

Where $H(S_i)$ and $H(T)$ are denotes the entropy values of the source and target projects, respectively. $k = 0$ corresponds to non-defective modules, and $k = 1$ corresponds to defective modules.

The entropy difference between the source and target projects is then calculated as Eqn. (8).

$$ED(S_i, T) = H(S_i) - H(T) \quad (8)$$

A smaller entropy difference indicates that the class distributions of the source and target projects are more consistent. This reduces the probability of biased learning and improves defect generalization capability during federated aggregation.

3.2.5. Wasserstein Distance

Features may overlap and balance of classes may be alike; however, projects may differ in the underlying distribution of metrics. Assume, $V_i(z)$ and $V_T(z)$ represent the cumulative distribution functions (CDFs) of metric z in S_i and T. The 1-Wasserstein distance is the cost of minimizing the cost to transform one distribution to another using Eqn. (9).

$$WD(S_i, T) = \int_0^1 |V_i^{-1}(z) - V_T^{-1}(z)| \quad (9)$$

Where $WD(S_i, T)$ means Wasserstein distance between source and target distributions.

$V_i^{-1}(z)$ and $V_T^{-1}(z)$ are denotes inverse cumulative distribution functions. Smaller $WD(S_i, T)$ means that source metrics are similar to the target as a statistical entity that reduces negative transfer.

The combination of three metrics into one relevance score τ_i for each source project is shown in Eqn. (10).

$$\tau_i = \omega_1 \cdot CS(T, S_i) - \omega_2 \cdot E(S_i, T) - \omega_3 \cdot WD(S_i, T) \quad (10)$$

Where ω_1, ω_2, and ω_3 are weighting coefficients as $\omega_1 + \omega_2 + \omega_3 = 1$. Positive CS adds to higher relevance. ED and WD are more severe on projects that do not add value.

3.3. Top-K Project Selection

Let $k \leq m$ be the number of top-ranked source projects. The selected set is defined as Eqn. (11).

$$S_{sel} = Top\ K(\{S_1, S_2, \dots, S_m\}, \tau_i) \quad (11)$$

These are the highest-ranking projects to be trained on local SOMRAGNN during federated training. This is the way to make sure that only the most appropriate sources impact the global model. There is a minimum threat of negative transfer. High-value sources are given priority in computational resources, which enhance the scale.

3.4. Local Model Training Using SOMRAGNN

After determining the highest-ranked source projects using the Ranking-Based mechanism, each of the selected source projects trains SOMRAGNN on its data. The SOMRAGNN test model is set up to learn the structural dependencies and interrelationships across modules in individual project software metrics graph efficiently and overcome the problems of over-smoothing, class imbalance, and project heterogeneity.

3.4.1. SOMRAGNN

The SOMRAGNN framework is a novel architecture that will allow to optimally predict heterogeneous cross-project defects through efficient integration of intricate structural dependencies and feature learning via adaptive strategies. The SOMRAGNN includes MRAGNN architecture which is a combination of multi-scale feature extraction, Residual Attention Module (ResAtnM) to capture the local and global contextual relationship of software project data. Furthermore, this model is integrating with HSBOA can facilitate fine-tuning of the model hyperparameters as well as feature weights and can be used to guarantee high levels of generalization in the case of varied source and target domains. The subsequent sections describe the core components of SOMRAGNN framework.

3.4.2. MRAGNN

In software defect prediction, it is necessary to know the complex structural interdependences between software modules to predict defect-prone components correctly. The software modules may be stored as nodes in a graph, and the relationships between modules, such as a call to a function, a hierarchy of inheritance, a shared variable or a dependency of access to a data structure, are the edges of the graph. This representation in the form of graphs reflects the topological and relation properties of software systems. Nevertheless, conventional deep learning systems assume that the independent features are represented as vectors without taking the relational nature of software entities into account. Such models are therefore likely to ignore the effect of module coupling, co-change behaviors, and context propagation through the software structure leading to a complete or biased feature learning in defect prediction. To overcome abovementioned issues, this study introduces MRAGNN, a graph-based model which combines multi-scale graph convolution, ResAtnM, and self-optimized learning parameters. This architecture represents intricate interdependencies among software modules at more than one level of abstraction, both local dependencies (between modules) and global dependencies (between the software system). The subsequent sections describe the architecture, mathematical design, and self-optimization mechanism used in MRAGNN.

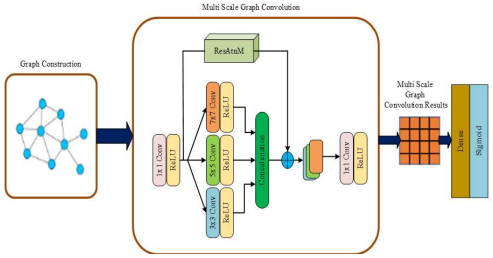

Figure 2: Architecture of MRAGNN

3.4.3. Graph Construction

A directed graph of source project S_i is constructed as $G_i = \{V_i, E_i, F_i\}$ that is taken to represent the structured and functional dependencies between software modules. This graph view enables the proposed SOMRAGNN framework to perform effective capturing of inter-module relationships to enable learning of both topological and metric-based features that are important to defect prediction.

The vertex $V_i = \{v_1, v_2, \ldots, v_{m_i}\}$ sets signify single software modules, with each node representing a particular module, class or file within the project. Each project has a number of nodes m_i which vary with the size and complexity of the project. Every node has the key software engineering properties that are the result of a static code analysis. All these quantitative measures describe the internal sub-structure of individual modules as well as their external dependencies to create a complete feature specification of predicting defects.

The directed dependencies between the modules were represented by the set of edges $E_i \subseteq V_i \times V_i$ captures. The edge $e_{xy} \in E_i$ has the value true in case the module v_x directly relies on or invokes the module v_y. This dependency could be based on calls of method, data interchange, inheritance. The directed character of the graph is that the direction of the influence between modules is maintained, the model can deduce asymmetric dependencies which are especially important in hierarchical or layered software architectures. The MRAGNN model uses this structure to learn the way the tendencies of defects spread across dependency chains within the project.

For each selected software project, a graph structure is constructed to explicitly model the dependency relationships among software modules. Let $G_i = (V_i, E_i, X_i)$denote the graph representation of the

i^{th}project, where V_i signifies set of software modules, $E_i$denotes dependency relationships among modules, and X_i denotes corresponding feature matrix. To mathematically encode these dependency relationships, an adjacency matrix $AM_i \in \{0,1\}^{m_i \times m_i}$ is defined, where m_i indicates total number of modules in project i. Each element of the adjacency matrix indicates whether a dependency relationship exists between two software modules. The adjacency relationship is formally defined as Eqn. (12).

$$AM_i(x,y) = \begin{cases} 1\,; if\ v_x\ depends\ on\ v_y \\ 0; otherwise \end{cases} \quad (12)$$

Where $AM_i(x,y)$ indicates dependency relation between modules v_x and v_y that represent software modules within project i.

This binary graph representation provides a compact and efficient description of software structural dependencies, enabling the proposed SOMRAGNN framework to perform graph-based message passing and convolutional learning effectively. The adjacency matrix is generated using software code parsing frameworks that extract function-call relationships, inheritance structures, package dependencies, and reference links among source files. However, directly using the raw adjacency matrix may lead to instability during graph convolution because nodes with large numbers of connections can dominate the information propagation process. Such imbalance may bias the learning mechanism and reduce the effectiveness of structural representation learning. To address this issue and improve numerical stability, a normalized symmetric adjacency matrix is computed before graph convolution operations. The normalized adjacency matrix is defined as Eqn. (13)

$$\widehat{AM}_i = \delta_i^{-\frac{1}{2}}\left(AM_i + I_{m_i}\right) \quad (13)$$

Where $\widehat{AM}_i$ means normalized symmetric adjacency matrix, δ_i signifies degree matrix of project i, and I_{m_i} denotes identity matrix used to introduce self-connections. The self-connections are added by I_{m_i}. To make sure that the characteristic of each module is reflected in its representation change, the self-loop $\left(AM_i + I_{m_i}\right)$ is added and makes sure that every module enhances its intrinsic traits during convolutional aggregation. The symmetric normalization $\delta_i^{-\frac{1}{2}}$ ensures that the flow of information through the graph is equally distributed, and the nodes with many connections do not dictate the learning process.

The structure of $\widehat{AM}_i$ is a normalized graph, which is the base representation of the SOMRAGNN model and allows efficient multi-scale passing of messages and aggregation of the information of software modules by attention. The resulting graph embedding therefore represents both structural topology as well as the metric-based semantics of the software project which provides a very strong base on cross project defect prediction.

3.4.4. Multi-Scale Graph Convolution

The MRAGNN model incorporates a multi-scale graph convolution to achieve good learning of both local structural dependencies between modules in software and the global topological effects on large projects overall. In contrast to the traditional single-scale graph convolutional networks that capture information across a range of immediate neighbors alone, the multi-scale architecture of MRAGNN allows the model to pass messages and propagate features across multiple receptive field scales thus improving the model ability to learn the hierarchical dependencies between codes as well as latent propagation defects between modules.

Each of the chosen projects S_i identified with its own normalized adjacency matrix $\widehat{AM}_i$. The multi-scale convolutional learning process operates at stages repeatedly with receptive field scales $k \in \{1,2,..,K\}$ representing increasingly larger aggregations of a neighbourhood or more distant dependencies. Such a design enables MRAGNN to jointly learn fine-grained intra-module connections as well as coarse-grained inter-module connections. At a specific convolution layer l and scale k, the node embedding update of project S_i is calculated as Eqn. (14).

$$\varepsilon_i^{(l,k)} = \sigma\left(\widehat{AM}_i^k \varepsilon_i^{l-1} \omega^{l,k}\right) \quad (14)$$

Where ε_i^{l-1} signifies the input node embeddings obtained from previous layer $l-1$. $\omega^{l,k}$ denotes learnable weight matrix specific to scale k, $\widehat{AM}_i^k$ indicates multi-hop normalized adjacency matrix, and $\sigma(\cdot)$ denotes Rectified Linear Unit (ReLU) non-linear activation function, which

introduces non-linearity and stabilizes the learning of complex relationships between features.

Eqn. (14) enables MRAGNN to learn hierarchical representations of each module, with small scales ($k = 1$) learning local dependencies and large scales ($k > 1$) learning high-level relationships in the project graph. This multi-level representation learning imitates the process of software defects spreading across the modular zone, and data makes the network predicts not only local fault signals but also global fault trends. Once the embeddings of all the scales have been obtained, MRAGNN aligns them by using multi-scale fusion mechanism and it is defined as Eqn. (15).

$$\varepsilon_i^{(l)} = \sum_{k=1}^{K} \alpha_k \varepsilon_i^{(l,k)} \quad (15)$$

Where $\varepsilon_i^{(l)}$ denotes fused embedding representation at l^{th} layer, and α_k represents adaptive scale coefficient of the scale k. The coefficient α_k satisfy the normalization constraint $\sum_{k=1}^{K} \alpha_k = 1$, which is used to make sure all the contribution of the receptive fields is balanced well. Such coefficients are not predetermined at all; they are optimized dynamically by means of HSBOA. Under this self-optimization, the algorithm dynamically updates the processes of the analysis of the fitness landscape -that is, giving preference to scales that add greater value to the accuracy of defect prediction.

The HSBOA-inspired scale weighting mechanism automatically represents the fact that the model will inherently learn the depth of the neighborhood, which will be the most significant contextual information to give a particular project. Indicatively, in the case of low-dimensional datasets shallow neighbourhoods can prevail as the dependency structures are simpler and in the case of high-dimensional datasets deeper multi-scale propagation is imperative to characterize intricate inter-module defect correlations.

3.4.5. Residual Attention Module

The Residual Attention Module (ResAtnM) is included in between the graph convolutional layers to improve the representational power of MRAGNN and enrich the over-smoothing issue that conventionally occurs in deep GNNs. The mechanism is applied to make sure important defect-related nodes are emphasized in representation without excessive loss of low-level structural and contextual information which is obtained bi-directionally by propagation of residual level.

Let $\varepsilon_i^{(l)} \in \mathbb{R}^{m \times d_l}$ represents the feature representation of node i at l^{th} layer, in which m represents the number of nodes and d_l signifies dimensionality of features at layer l. The ResAtnM adds an attention-based transformation that concentrates more on informative nodes intelligently and this can be formulated as Eqn. (16).

$$Y_i^l = SoftMax\left(\frac{\left(\varepsilon_i^{(l)} W_Q\right)\left(\varepsilon_i^{(l)} W_K\right)^{\top}}{\sqrt{d_k}}\right)\varepsilon_i^{(l)} W_V \quad (16)$$

Where W_K, W_Q, and W_V key, query, and value weight matrix respectively. d_k represents dimensionality of attention space, and $SoftMax$ function normalizes the attention scores to confirm a probabilistic interpretation of importance weights across nodes.

The self-attention mechanism is also the inspiration behind this formulation, where compatibility between query and key embeddings is used to determine the level of concentration that should be placed on the neighbors of each node. Scaling the dot product by $\sqrt{d_k}$ is the force that by preventing extreme values of gradients when training the model, it provides stable learning dynamics.

The following step is to avoid degradation of features that come about due to nonlinear transformations, whereby a residual propagation strategy is used. The attention integration of the residual attention feature fuses the original feature of the nodes with the attention-wise output, which is given as Eqn. (17).

$$\hat{\varepsilon}_i^{(l)} = \varepsilon_i^{(l)} + \gamma Y_i^l \quad (17)$$

Where γ is a parameter to be learned that is used to adjust the contribution of the feature being led to by the attention, in comparison to the underlying features. A higher value focuses on attention refinement whereas a lower value focuses on original feature retention. This residual connection helps in cancelling out vanishing gradient problems and conserving rich local structural information, and hence does not lose

discriminative representations in deeper architectures. Learning is also better stabilized as the ResAtnM enables backpropagation of the gradient using residual and attention pathways, such that the model captures low-level dependencies among the nodes in a graph and high-level relational patterns of the layers of the graph. Lastly, the aggregation of the residual-attention-enhanced-features of multi-scales on layers L generate the final graph-level embedding which is defined as Eqn. (18).

$$\hat{\varepsilon}_i^{(L)} = Concat\left(\hat{\varepsilon}_i^{(1)}, \hat{\varepsilon}_i^{(2)}, \dots, \hat{\varepsilon}_i^{(L)}\right) \quad (18)$$

Where $Concat$ is a readout function that supported as concatenation, mean pooling or global attention pooling, to combine the representations of all the hierarchical features. This operation combines both the shallow and deep semantic features and enables the model to maintain the granularity of nodes and also to encounter the overall structural context of the graph of software projects. Figure 3. shows the Architecture of Residual Attention Module

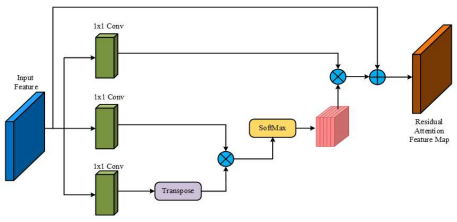

Figure 3.: Architecture of Residual Attention Module

3.4.6. Output Layer

After residual attention, last stage in SOMRAGNN architecture is to transform the learned node embeddings into the probability of defect prediction used by a fully connected output classification layer. This layer performs the mapping of the high-dimensional latent of every software module to a binary prediction which tells whether the software module is defect-prone (1) or clean (0).

Let $\hat{\varepsilon}_{ij}^{(L)}$ indicate the final embedding of node j in project i at the L layers of the residual attention and graph convolutional transformations. The probability $\hat{P}_{ij}$ of node j in the defect-prone class is estimated to be the predicted probability of the node being in the defect-prone using Eqn. (19).

$$\hat{P}_{ij} = \sigma\left(\hat{\varepsilon}_{ij}^{(L)} W_0 + b_0\right) \quad (19)$$

Where W_0 represents output weight matrix that linearly maps the final feature embedding to prediction space, b_0 denotes bias term that allows the model to adaptively adjust the decision boundary, and $\sigma(\cdot)$ indicates sigmoid activation function. The sigmoid activation is especially appropriate in binary classification problems such as defect prediction, where the model is used to produce a continuous probability, which can be threshold (usually at 0.5) to produce the final binary solution.

3.4.7. Loss Function

SOMRAGNN utilizes Weighted Binary Cross-Entropy (WBCE) loss function to handle the class imbalance that exists in datasets of defect prediction. This method increases the sensitivity to cases that are prone to defects by penalizing the misclassification of minority class samples more severely. Project i's loss function is described as Eqn. (20).

$$F(L) = -\frac{1}{m_i}\sum_{j=1}^{m_i}\left[\omega_1 P_{ij}\log\left(\hat{P}_{ij}\right) + \omega_2\left(1 - P_{ij}\right)\log\left(1 - \hat{P}_{ij}\right)\right] \quad (20)$$

Where m_i denotes the total number of modules in project i, $P_{ij} \in \{0,1\}$ signifies ground truth label of module j, $\hat{P}_{ij}$ indicates predicted possibility of being defective, ω_1 and ω_2 are class weighting factors equivalent to defect-prone and defect-free classes, respectively.

The proposed SOMRAGNN framework includes a self-optimization mechanism where its key hyperparameters including attention scaling factor, learning rate, number of convolutional scales, and residual depth dynamically tuned using HSBOA.

3.4.8. HSBOA

The study introduces HSBOA to improve the performance of SOMRAGNN in relation to different projects. The HSBOA is a metaheuristic that aims to self-optimize the attention parameters and the hyperparameters of the learning process of the SOMRAGNN. The optimization targets to maximize detection of defects and reduce the overfitting tendency and generate generalization across a heterogeneous project data. In HSBOA, SOMRAGNN parameter sets consisting of attention

weights, learning rates, residual scaling factors, and convolutional filter coefficients are represented by each candidate solution (bobcat). Assume that the size of the search space is n, the number of parameters to be optimized, and the population size is M. The population matrix is represented as Eqn. (21).

$$S = \begin{bmatrix} S_1 \\ \vdots \\ S_i \\ \vdots \\ S_M \end{bmatrix}_{M \times n} \quad ; \quad S_i = [s_{i,1}, s_{i,2}, \dots, s_{i,n}] \tag{21}$$

Where $s_{i,d}$ represents d^{th} SOMRAGNN parameter of i^{th} candidate solution. Initial locations are randomly created within predefined parameter bounds as shown in Eqn. (22).

$$s_{i,d} = LB_d + Rand \cdot (UB_d - LB_d) \tag{22}$$

Where $Rand$ denotes uniform random number in [0,1]. UB_d and LB_d are upper and lower bounds of d^{th} parameter respectively.

3.4.9. Hybrid Exploration Phase: Secretary Bird-Bobcat Integration

During the exploration phase, HSBOA combines the BOA candidate-prey movement with the three-stage hunting plan of the Secretary Bird, which guarantees efficient world-wide search. The SBO algorithm divided the entire hunting process into three equal time intervals, namely $t < \frac{1}{3}T$, $\frac{1}{3}T < t < \frac{2}{3}T$ and $\frac{2}{3}T < t < T$, corresponding to the three phases of the secretary bird's predation: searching for prey, consuming prey, and attacking prey. The candidate prey set of i^{th} bobcat is defined as Eqn. (23).

$$LP_i = \{S_k | F_k < F_i\}, k = 1,2, \dots, M \tag{23}$$

Where $F_i = F(S_i)$ denotes the value of objective function, which is a weighted average of accuracy, F1 -score, and False Positive Rate (FPR) as shown in Eqn. (24).

$$F(S_i) = \alpha_1 \cdot (1 - Accuracy) + \alpha_2 \cdot (1 - F1\ score) + \alpha_3 \cdot (FPR) \tag{24}$$

Where α_1, α_2, and α_3 are weighted parameters such as $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

A bobcat randomly selects a prey $SP_i \in LP_i$. The hybrid position update is achieved as Eqn. (25).

$$H(s_{i,j}) = SBO(s_{i,j}) + \beta \cdot \left(BOA(s_{i,j}) - SBO(s_{i,j})\right) \tag{25}$$

Where $SBO(s_{i,j})$ denotes current position of i^{th} individual in j^{th} dimension, obtained purely from the SBO update using one of the three stages such as searching, consuming, and attacking prey. $BOA(s_{i,j})$ represents guidance position derived from BOA update mechanism. β indicates hybrid influence coefficient $(0 \le \beta \le 1)$ that regulating the proportion of BOA-based refinement injected into SBO update. It is typically drawn from a uniform random distribution or adaptively reduced over time. Additionally, t signifies current iteration number, while T denotes maximum number of optimization iterations. These parameters are used to regulate the transition among the different hunting phases of HSBOA optimization strategy.

Stage 1: Searching for Prey $\left(t < \frac{1}{3}T\right)$

This preliminary step entails the use of Differential Evolution (DE)-based mutation to introduce diversity to prevent local optima as shown in Eqn. (26).

$$s_{i,j}^N = s_{i,j} + (s_{r1} - s_{r2}) \times R_1 \tag{26}$$

Where s_{r1} and s_{r2} are randomly selected population members, R_1 defines randomly created array of dimension $1 \times Dim$ from the interval [0, 1], where Dim indicates dimensionality of solution space. The update is accepted if it recovers fitness as represented in Eqn. (27).

$$S_i = \begin{cases} s_{i,j}^N, F(s_i^N) < F(S_i) \\ S_i\ , otherwise \end{cases} \tag{27}$$

Where $s_{i,j}^N$ signifies its value of j^{th} dimension, and $F(s_i^N)$ denotes its fitness value of objective function.

Stage 2: Consuming Prey $\left(\frac{1}{3}T < t < \frac{2}{3}T\right)$

During this step, exploration is made more precise with Brownian motion controlled stochastic perturbations as shown in Eqn. (28).

$$s_{i,j}^{N} = s_{best,j} + exp\left(\left(\frac{t}{T}\right)^{4}\right) \cdot (RB - 0.5) \cdot \left(s_{best,j} - s_{i,j}\right) \quad (28)$$

Where s_{best} current best solution, and RB signifies Brownian random variable. Updates are acknowledgeable if they improve the objective.

Stage 3: Attacking Prey $\left(\frac{2}{3}T < t\right)$

Lastly, long jumps with global search space exploration are possible with Levy flight using Eqn. (29).

$$s_{i,j}^{N} = s_{best,j} + \left(\left(1 - \frac{t}{T}\right) \wedge \left(2 \cdot \frac{t}{T}\right)\right) \cdot s_{i,j} \cdot L_D \quad (29)$$

Where L_D represents Levy flight that enhances the optimization accuracy of algorithm.

$$L_D = 0.5 \times Levy(Dim) \quad (30)$$

Where $Levy(Dim)$ denotes Levy flight distribution function. It is determined using Eqn. (31).

$$Levy(Dim) = c \times \frac{p \times \sigma}{|q|^{\frac{1}{\eta}}} \quad (31)$$

Where c denotes fixed constant of 0.01, η indicates denotes fixed constant of 1.5, p and q are stands for random numbers in the interval [0, 1]. The mathematical expression of σ is defined as Eqn. (32).

$$\sigma = \left[\frac{\Gamma(1+\eta) \times sin\left(\frac{\pi\eta}{2}\right)}{\Gamma\left(\frac{1+\eta}{2}\right) \times \eta \times 2\left(\frac{\eta-1}{2}\right)}\right]^{\frac{1}{\eta}} \quad (32)$$

Where Γ signifies the gamma function.

3.4.10. Exploitation Phase: Bobcat Chasing Prey

Succeeding exploration, the exploitation stage enhances local search using standard BOA chase approach using Eqn. (33).

$$s_{i,j}^{P} = s_{i,j} + \frac{1-2R_{i,j}}{1+t} \cdot s_{i,j} \quad (33)$$

$$S_i = \begin{cases} S_i^P, F(S_i^N) < F(S_i) \\ S_i, otherwise \end{cases} \quad (34)$$

Where, S_i^P represents objective function value, $R_{i,j}$ indicates random values from the interval [0, 1], t denotes iteration counter, $F(S_i^N)$ indicates new position determined for i^{th} bobcat based on the exploitation phase of BOA, and $s_{i,j}^P$ is its j^{th} dimension. This stage confirms local convergence close to promising solutions while evading overshooting.

Based on iteration input, HSBOA adaptively adjusts the population size, step-size factors, and switching probability ρ between BOA and SBO exploration as expression shown in Eqn. (35).

$$\rho(t) = \rho_{min} + (\rho_{max} - \rho_{min}) \cdot \left(\frac{t}{T}\right) \quad (35)$$

This adaptive parameter tuning ensures a smooth transition from global exploration to local exploitation.

3.5. FedRWAvg

The next major step in the federated learning process is combining the locally learned models into a single global model when local sources projects have completed their training phases using SOMRAGNN model. Under the classic federated learning approaches like FedAvg, the averaging of the entire local model parameters is generally conducted based on the dataset sizes. But this simple averaging strategy overlooks relevance, data quality or structural similarity of various source projects that might cause knowledge dilution or transfer negative in HCPDP. To address these issues, the study proposes FedRWAvg mechanism that integrates the project ranking scores to determine the process of aggregation. The projects that are more similar and related to the target project must have an influence on the global parameter update thus the less relevant ones are proportionally less.

Mathematically, let θ_i signify the local model parameters estimated on i^{th} source project S_i, and R_i indicate the ranking score of that project the similarity metrics of the features, defect distribution alignment and transferability measures calculated in

Ranking-Based Source Project Selection. Those global model parameters θ_g are calculated as a weighted sum of local parameters, which are defined by Eqn. (36).

$$\theta_g = \frac{\sum_{i=1}^{v} R_i \cdot \theta_i}{\sum_{i=1}^{v} R_i} \quad (36)$$

Where θ_g denotes aggregated global parameter vector shared across all clients, v indicates top-ranked source projects designated for participation in the federated update, R_i indicates ranking weight of project S_i, and θ_i stands for locally optimized model parameters at client i.

The scale of R_i itself is obtained as a result of multi-feature relevance function that applies statistical feature alignment, distance between distance and defects, and similarity of representation between the source and target project. In this way, the aggregation focuses on semantically related and structurally similar projects of the source domain to the target one, and thus, the generalization is enhanced. The weighted averaging process can assure that the federated global model can utilize better the insights of the most relevant source projects, and the negative impact of irrelevant or low-quality datasets is reduced. Mathematically, the update rule can also be considered the objective function that expressed in Eqn. (37).

$$\min_{\theta_g} \sum_{i=1}^{v} R_i \left\| \theta_g - \theta_i \right\|^2 \quad (37)$$

Eqn. (37) defined to determine a global parameter set that minimizes the weighted error of all the involved local models. Besides, at the end of every communication round t, the global model parameters θ_g^t are redistributed to all the chosen clients to continue in successive training cycles. This process of successive round-based rank-based aggregation allows the global model to adaptively adopt an ideal representation of local structural variations as well as cross-project general trends. Essentially, the FedRWAvg system is a suitable addition to classical federated optimization, which incorporates intelligent relevance weighting, leading to excellent knowledge transfer, resilience towards heterogeneity, and consistent performance in HCPDP contexts.

4. EXPERIMENTAL SETUP

4.1. Dataset description

This study used five popular, publicly available software defect datasets including PROMISE, SOFTLAB, NASA (MDP), AEEEM, and ReLink, all of which are widely used to provide a complete and thorough assessment of the proposed framework. The data sets together offer a benchmark environment for making heterogeneous cross-project defect prediction models, since the combinations of project size, dimensionality of features, and distribution of defects vary across the data sets.

The PROMISE [33] data set comprises projects of medium scale (205 to 965) comprising 20 software metrics for each project and each module for the seven medium scale Java projects, as presented in the Table. It is reasonable to have a moderate imbalance of classes for evaluation of cross-project learning with a distribution of defects ranging from 6.10% to 25.40%.

The SOFTLAB [34] dataset is a collection of five projects that have a regular structure, the same number of metrics per project (40) and the same number of modules per project (160). The defect percentages vary between 16.70% and 20.30% and the data is relatively homogeneous which makes it useful to assess the stability under controlled conditions.

The NASA (MDP) [35] dataset is a large-scale industrial standard set and spans a very broad range of project sizes from 403 to 10,885 modules. This contains 21 software measurements per job and defect rate ranging from 6.90% to 19.30%, with a high amount of variation not only in the amount but also in the distribution of metrics.

AEEEM [36] is a high-dimensional dataset comprising seven software systems based on eclipse, and 61 metrics per project. The number of modules varies between 324 and 2,107 and the defect rates are between 11.60% and 21.00% making it ideal for assessing complex CPDP scenarios.

Lastly, there are seven mobile and Apache-based systems in the ReLink dataset [37] that have 26 metrics per project. It also exhibits strong class imbalance (defect rates from 10.90% to 27.10%) which can serve as a difficult standard to assess robustness in the face of skewed distribution.

Table 1: Statistical characteristics of all datasets

Dataset	No. of Projects	Total Modules	No. of Metrics	Defect Rate Range (%)
PROMISE	7	205 – 965	20	6.10 – 25.40
SOFTLAB	5	160 – 160	40	16.70 – 20.30
NASA (MDP)	7	403 – 10,885	21	6.90 – 19.30
AEEEM	7	324 – 2,107	61	11.60 – 21.00
ReLink	7	321 – 796	26	10.90 – 27.10

4.2. Model training

A stratified experimental protocol is used for all datasets in order to obtain a fair and repeatable assessment of the proposed framework. The data partitioning strategy ensures intra-project distribution and inter-project generalization capability, as the study is a federated learning study with heterogeneous cross-project defect prediction. Data leakage between federated clients is avoided by partitioning each source project at the project level. All the data is segmented by a stratified random sampling method, which preserves the proportion of defective modules and non-defective modules in each subset. This is particularly important due to the inherent class imbalance present in software defect datasets. The splitting is carried out in the following manner: 70% is used for training, 15% for validation and 15% for test. The target project is dealt with differently depending on CPDP evaluation settings. It is not used in training, but only at the end for the global aggregated model evaluation and test. In the federated scenario, the selected source projects are independent clients. Local SOMRAGNN model is trained per client with its local training subset. The data used for hyperparameter tuning and HSBOA-based attention optimization is called validation data while the rest is for local performance estimation prior to aggregation. Following local training, only the model parameters are sent to the central server, guaranteeing privacy preservation in compliance with the principles of federated learning. The same data split process is used for all baseline models to avoid evaluation bias, and avoid issues with data splitting. Furthermore, every model is tested on the same target project test set that is not seen during training. Table 2 summarizes the hyperparameters of model.

Table 2: Hyperparameters of proposed model

Parameter	**Value**
Optimizer	Adam
Learning Rate	0.001
Batch Size	32
Epochs	100
Number of Client	3
Top-K Selection (Ranked source projects)	3
Communication Rounds	10
Population Size (M) for HSBOA	30
No. of Iterations (T)	50

4.3. Performance metrics

The evaluation measures such as Accuracy, Specificity, Precision, Recall, F1-score, and False Positive Rate (FPR) are used to evaluate the performance of proposed model. Table 3 shows the summary List of Evaluation Metrics.

Table 3: List of Evaluation Metrics

Metric	**Description**
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Precision	$\frac{TP}{TP + FP}$

Recall	$\frac{TP}{TP + FN}$
F1-Score	$2 * \frac{precision * recall}{precision + recall}$
FPR	$\frac{FP}{FP + TN}$

5. RESULTS AND DISCUSSIONS

This section presents the experimental outcomes obtained from multiple benchmark datasets, including NASA, PROMISE, AEEEM, ReLink, and SOFTLAB. The results highlight the superior performance of the proposed model compared to existing methods, demonstrating enhanced accuracy, precision, recall, and F1-score with consistently lower false positive rates. The proposed model was evaluated against various current state of the art models including RH-CPDP [25], TriStage-CPDP [26], SCAG-LSTM [29], and ECFML [32] to verify the performance of proposed framework.

5.1. Performance analysis of proposed and baseline models across datasets

The experimental results with the PROMISE dataset in Table 3 show that the proposed framework is effective in the Top-K ranked source selection. The proposed model achieves the best performance in all the target projects, outperforming all the baseline models. The improvement is especially noticeable in Accuracy, F1-score and FPR which shows the improvement in predictive power and class discrimination. For example, in the JEdit target and POI target, the proposed model achieves the accuracy value of 0.968 and 0.969 respectively, which is about 1.5–2% higher than that of the best baseline RH-CPDP. These patterns are seen for all considered metrics, which further validates the effectiveness of the federated ranking-guided learning strategy. The model's FPR consistently decreases to as low as 0.015–0.018, further signifying its ability to accurately reduce the number of misclassifications of un-defective modules, which is a crucial point in software defect prediction. This performance can be explained based on three important aspects of the proposed framework. The ranking-based Top-K selection guarantees only relevant source projects are used for knowledge transfer, minimizing negative transfer effects. Second, the SOMRAGNN architecture is able to well capture structural relationships among software modules with multi-scale graph convolution and residual attention mechanisms. Third, the optimization based on the HSBOA improves the tuning of the parameters, which can then be adapted across the source domain that are heterogeneous. The PROMISE dataset results confirm the proposed framework, which provides stable and high performance cross-project defect prediction with good generalizability on heterogeneous software systems.

Table 3: Performance analysis of proposed and baseline models on PROMISE dataset

Source Projects → Target	Measure	TriStage-CPDP	ECFML	SCAG-LSTM	RH-CPDP	Proposed
{POI, Camel, Lucene} → JEdit	Accuracy	0.912	0.926	0.941	0.953	**0.968**
	Precision	0.908	0.923	0.938	0.951	**0.967**
	Recall	0.909	0.924	0.939	0.952	**0.967**
	F1-Score	0.908	0.923	0.938	0.951	**0.967**
	FPR	0.071	0.058	0.043	0.031	**0.018**
{Camel, Lucene, Ant} → POI	Accuracy	0.914	0.928	0.943	0.955	**0.969**
	Precision	0.91	0.925	0.94	0.953	**0.968**
	Recall	0.911	0.926	0.941	0.954	**0.968**
	F1-Score	0.91	0.925	0.94	0.953	**0.968**
	FPR	0.069	0.056	0.041	0.029	**0.017**
{Lucene, POI,	Accuracy	0.916	0.931	0.945	0.957	**0.971**

Ant} → Camel	Precision	0.913	0.928	0.943	0.955	**0.97**
	Recall	0.914	0.929	0.944	0.956	**0.97**
	F1-Score	0.913	0.928	0.943	0.955	**0.97**
	FPR	0.067	0.054	0.039	0.027	**0.016**
{Ant, POI, Camel} → Lucene	Accuracy	0.918	0.933	0.947	0.959	**0.973**
	Precision	0.915	0.93	0.945	0.957	**0.972**
	Recall	0.916	0.931	0.946	0.958	**0.972**
	F1-Score	0.915	0.93	0.945	0.957	**0.972**
	FPR	0.065	0.052	0.037	0.025	**0.015**
{Camel, Lucene, Velocity} → Ant	Accuracy	0.915	0.93	0.944	0.956	**0.97**
	Precision	0.911	0.927	0.942	0.954	**0.969**
	Recall	0.912	0.928	0.943	0.955	**0.969**
	F1-Score	0.911	0.927	0.942	0.954	**0.969**
	FPR	0.068	0.055	0.04	0.028	**0.017**

The experimental results with the SOFTLAB data set reported in Table 4 demonstrate the robustness and generalization capacity of proposed framework in structurally diverse but relatively homogeneous software environment. SOFTLAB is composed of projects, which have uniform feature dimensionality of 40 features and moderate heterogeneity in defect distribution, well controlled, therefore, suitable for cross-project transferability evaluations under controlled heterogeneity. This proposed model consistently delivers the best results on all the measured parameter with accuracy of 0.970-0.978 and F1-scores of above 0.970 for all the various combinations. The results are clearly better than the baseline models with improvements in accuracy of 1.2–2.5% and a significant drop in FPR to 0.011–0.014. The selection of Top-K ranked source projects by proposed framework leads to effective federated training results. This helps to minimize noise from less informative sources and improves the efficiency of knowledge transfer. In addition, multi-scale graph convolution and residual attention mechanisms are used in the SOMRAGNN to learn local and global dependencies so that the model can accurately predict the defect position at the module level. The HSBOA-based optimization mechanism is also designed to dynamically adjust attention and learning parameters to ensure stability of converging results over all the federated clients, further improving model performance. The overall results of the SOFTLAB indicate that the proposed approach represents a viable approach for real-world software engineering scenarios, as it is both scalable and practical, and can obtain good performance and stability performance results in moderately homogeneous datasets without compromising accuracy.

Table 4: Performance analysis of proposed and baseline models on SOFTLAB dataset

Source Projects → Target	Measure	TriStage-CPDP	ECFML	SCAG-LSTM	RH-CPDP	Proposed
{POI, Camel, Lucene} → JEdit	Accuracy	0.918	0.932	0.947	0.958	**0.972**
	Precision	0.914	0.929	0.944	0.956	**0.971**
	Recall	0.915	0.93	0.945	0.956	**0.971**
	F1-Score	0.914	0.929	0.944	0.956	**0.971**
	FPR	0.066	0.053	0.039	0.027	**0.014**
{Camel, Luce	Accuracy	0.92	0.934	0.949	0.96	**0.974**

ne, Ant} → POI	Precision	0.917	0.931	0.947	0.958	**0.973**
	Recall	0.918	0.932	0.948	0.959	**0.973**
	F1-Score	0.917	0.931	0.947	0.958	**0.973**
	FPR	0.064	0.051	0.037	0.025	**0.013**
{Lucene, POI, Ant} → Camel	Accuracy	0.917	0.931	0.946	0.957	**0.971**
	Precision	0.913	0.928	0.944	0.955	**0.97**
	Recall	0.914	0.929	0.945	0.956	**0.97**
	F1-Score	0.913	0.928	0.944	0.955	**0.97**
	FPR	0.067	0.054	0.04	0.028	**0.015**
{Ant, POI, Camel} → Lucene	Accuracy	0.922	0.936	0.951	0.962	**0.976**
	Precision	0.919	0.933	0.949	0.96	**0.975**
	Recall	0.92	0.934	0.95	0.961	**0.975**
	F1-Score	0.919	0.933	0.949	0.96	**0.975**
	FPR	0.062	0.049	0.035	0.023	**0.012**
{Camel, Lucene, Velocity} → Ant	Accuracy	0.924	0.938	0.953	0.964	**0.978**
	Precision	0.921	0.935	0.951	0.962	**0.977**
	Recall	0.922	0.936	0.952	0.963	**0.977**
	F1-Score	0.921	0.935	0.951	0.962	**0.977**
	FPR	0.06	0.047	0.033	0.021	**0.011**

The proposed framework is shown to be effective across a heterogeneous cross-project defect prediction setting in the experimental evaluation on the NASA (MDP) dataset. The results clearly show that knowledge transfer across projects is degrading the performance of all models as a consequence of distributional divergence and class imbalance problems, and the proposed approach consistently outperforms all the baseline approaches. As for baseline models, RH-CPDP is a relatively competitive model, which suggests that hybrid representation learning is useful for complex tasks in CPDP. However, baseline models are still constrained from dealing with extreme distribution shifts across projects. The proposed framework outperforms other frameworks in all the evaluation measure, the accuracy value is 0.958 to 0.965 and the accuracy value is higher and the precision, recall and F1 score is higher compared with others. The improvement is achieved due to three aspects of ranking-based Top-K source selection effectively mitigates negative transfer by discarding the irrelevant projects; federated aggregation based on FedRWAvg only aggregates highly relevant knowledge, as the local model is used to weight the contributions of the selected projects; and SOMRAGNN with HSBOA optimization improves the feature representation learning by using multi-scale graph convolution and adaptive attention mechanism to select the relevant features. Moreover, the improvement in the number of false positives in all target projects indicates the strength of the proposed model to ensure that a minimum number of non-defective modules are misclassified. In general, it is confirmed that the combination of ranking guided federated learning and graph based deep learning is a valuable approach for improving generalization in heterogeneous CPDP settings.

Table 5: Performance analysis of proposed and baseline models on NASA (MDP) dataset

Source Projects → Target	Measure	TriStage-CPDP	ECFML	SCAG-LSTM	RH-CPDP	Proposed
{POI, Camel,	Accuracy	0.901	0.915	0.929	0.942	**0.961**

Lucene} → JEdit	Precision	0.897	0.912	0.926	0.94	**0.96**
	Recall	0.898	0.913	0.927	0.941	**0.96**
	F1-Score	0.897	0.912	0.926	0.94	**0.96**
	FPR	0.082	0.069	0.055	0.041	**0.022**
{Camel, Lucene, Ant} → POI	Accuracy	0.903	0.917	0.931	0.944	**0.963**
	Precision	0.9	0.914	0.928	0.942	**0.962**
	Recall	0.901	0.915	0.929	0.943	**0.962**
	F1-Score	0.9	0.914	0.928	0.942	**0.962**
	FPR	0.079	0.066	0.052	0.038	**0.02**
{Lucene, POI, Ant} → Camel	Accuracy	0.906	0.92	0.934	0.946	**0.965**
	Precision	0.903	0.917	0.931	0.944	**0.964**
	Recall	0.904	0.918	0.932	0.945	**0.964**
	F1-Score	0.903	0.917	0.931	0.944	**0.964**
	FPR	0.076	0.063	0.049	0.036	**0.019**
{Ant, POI, Camel} → Lucene	Accuracy	0.899	0.913	0.927	0.94	**0.958**
	Precision	0.895	0.91	0.924	0.938	**0.957**
	Recall	0.896	0.911	0.925	0.939	**0.957**
	F1-Score	0.895	0.91	0.924	0.938	**0.957**
	FPR	0.084	0.071	0.057	0.043	**0.024**
{Camel, Lucene, Velocity} → Ant	Accuracy	0.901	0.915	0.929	0.942	**0.961**
	Precision	0.897	0.912	0.926	0.94	**0.96**
	Recall	0.898	0.913	0.927	0.941	**0.96**
	F1-Score	0.897	0.912	0.926	0.94	**0.96**
	FPR	0.082	0.069	0.055	0.041	**0.022**

The experimental evaluation on the AEEEM dataset depicted in Table 6 shows that the proposed framework can be effective in solving the problems of heterogeneous cross-project defect prediction in the complex and high-dimensional feature spaces. The dataset is the collection of multiple eclipse-based projects, each having 61 software metrics and defect distributions from 11.6% to 21%, which makes it a difficult set of projects to use for cross-project generalization. The results demonstrate that traditional CPDP approaches like TriStage-CPDP exhibit modest performance since they cannot properly capture inter-project heterogeneity. However, the proposed FedHCPDP framework is consistently more accurate over all of the target projects (accuracy values between 0.970 and 0.976). The three main steps including ranking based source selection mechanism, which is able to effectively filter out irrelevant source projects and mitigate negative transfer; the FedRWAvg, which guarantees that only highly relevant local models are significantly impacting the global model; and (iii) the SOMRAGNN architecture optimized with HSBOA, which allows to learn multi-scale features and adapt attention to the diverse graph structures. In addition, the FPR results obtained by the proposed method are always smaller than the results obtained by the other methods, showing the effectiveness of the method to reduce false positive classification of non-defective modules, an important aspect in software defect prediction systems. The results confirm that ranking-guided federated learning combined with multi-scale graph neural networks is a very effective approach for enhancing the generalization in a heterogeneous software engineering environment.

Table 6: Performance analysis of proposed and baseline models on AEEM dataset

Source Projects → Target	Measure	TriStage-CPDP	ECFML	SCAG-LSTM	RH-CPDP	Proposed
{POI, Camel, Lucene} → JEdit	Accuracy	0.916	0.93	0.944	0.957	**0.973**
	Precision	0.912	0.927	0.941	0.955	**0.972**
	Recall	0.913	0.928	0.942	0.956	**0.972**
	F1-Score	0.912	0.927	0.941	0.955	**0.972**
	FPR	0.07	0.056	0.042	0.028	**0.015**
{Camel, Lucene, Ant} → POI	Accuracy	0.919	0.933	0.947	0.959	**0.975**
	Precision	0.915	0.93	0.944	0.957	**0.974**
	Recall	0.916	0.931	0.945	0.958	**0.974**
	F1-Score	0.915	0.93	0.944	0.957	**0.974**
	FPR	0.068	0.054	0.04	0.026	**0.014**
{Lucene, POI, Ant} → Camel	Accuracy	0.914	0.928	0.942	0.955	**0.971**
	Precision	0.91	0.925	0.939	0.953	**0.97**
	Recall	0.911	0.926	0.94	0.954	**0.97**
	F1-Score	0.91	0.925	0.939	0.953	**0.97**
	FPR	0.071	0.057	0.043	0.029	**0.016**
{Ant, POI, Camel} → Lucene	Accuracy	0.921	0.935	0.949	0.961	**0.976**
	Precision	0.917	0.932	0.946	0.958	**0.975**
	Recall	0.918	0.933	0.947	0.959	**0.975**
	F1-Score	0.917	0.932	0.946	0.958	**0.975**
	FPR	0.066	0.052	0.038	0.024	**0.013**
{Camel, Lucene, Velocity} → Ant	Accuracy	0.918	0.932	0.946	0.958	**0.972**
	Precision	0.914	0.929	0.943	0.956	**0.971**
	Recall	0.915	0.93	0.944	0.957	**0.971**
	F1-Score	0.914	0.929	0.943	0.956	**0.971**
	FPR	0.069	0.055	0.041	0.027	**0.015**

The experiment results on the ReLink data set reported in Table 7 show the robustness and generality of the proposed FedHCPDP framework in the heterogeneous cross-project defect prediction scenario. By comparison, the proposed FedHCPDP framework always performs better on all the target projects with accuracy ranging from 0.965 to 0.970. This improvement is mainly due to the main contributions including Top-K ranking-based source project selection mechanism, FedRWAvg strategy, and HSBOA. Further, the low FPR obtained in all the cases in the proposed framework suggest that the detection ability is enhanced and the number of false positives (non-defective modules) is reduced. In general, the results support the hypothesis that the combination of ranking-aware federated learning and graph-based deep learning is very beneficial for prediction in highly heterogeneous software engineering environments like ReLink.

Table 7: Performance analysis of proposed and baseline models on ReLink dataset

Source Projects → Target	Measure	TriStage-CPDP	ECFML	SCAG-LSTM	RH-CPDP	Proposed
{POI, Camel, Lucene} → JEdit	Accuracy	0.909	0.923	0.938	0.951	**0.967**
	Precision	0.905	0.92	0.935	0.948	**0.966**
	Recall	0.906	0.921	0.936	0.949	**0.966**
	F1-Score	0.905	0.92	0.935	0.948	**0.966**
	FPR	0.072	0.058	0.044	0.03	**0.016**
{Camel, Lucene, Ant} → POI	Accuracy	0.912	0.926	0.941	0.954	**0.969**
	Precision	0.908	0.923	0.938	0.951	**0.968**
	Recall	0.909	0.924	0.939	0.952	**0.968**
	F1-Score	0.908	0.923	0.938	0.951	**0.968**
	FPR	0.07	0.056	0.042	0.028	**0.015**
{Lucene, POI, Ant} → Camel	Accuracy	0.91	0.924	0.939	0.952	**0.966**
	Precision	0.906	0.921	0.936	0.949	**0.965**
	Recall	0.907	0.922	0.937	0.95	**0.965**
	F1-Score	0.906	0.921	0.936	0.949	**0.965**
	FPR	0.073	0.059	0.045	0.031	**0.017**
{Ant, POI, Camel} → Lucene	Accuracy	0.913	0.927	0.942	0.955	**0.97**
	Precision	0.909	0.924	0.939	0.952	**0.969**
	Recall	0.91	0.925	0.94	0.953	**0.969**
	F1-Score	0.909	0.924	0.939	0.952	**0.969**
	FPR	0.069	0.055	0.041	0.027	**0.014**
{Camel, Lucene, Velocity} → Ant	Accuracy	0.911	0.925	0.94	0.953	**0.968**
	Precision	0.907	0.922	0.937	0.95	**0.967**
	Recall	0.908	0.923	0.938	0.951	**0.967**
	F1-Score	0.907	0.922	0.937	0.95	**0.967**
	FPR	0.071	0.057	0.043	0.029	**0.015**

5.2. Ablation study

The results of the ablation study on AEEEM dataset using the JDT as a target project depicted in Table 8 give a clear quantitative justification for the contribution of each component in the proposed FedHCPDP framework. The results show that the full model achieves the best performance on all the evaluation metrics including accuracy of 0.973 and F1 score of 0.972, which validates the effectiveness of the integrated architecture that integrates ranking based source selection, federated aggregation and graph based deep learning. It can be seen that the performance of FedRWAvg is significantly degraded when it is replaced by the widely used FedAvg aggregation strategy which accuracy decreases from 0.973 to 0.956. This decrease serves as a reminder of the need for ranking guided weighting during federated aggregation. In contrast to FedAvg, FedRWAvg gives more influence to a more relevant source project according to similarity-driven ranking scores. This mechanism can effectively suppress negative transfer from less

relevant domains, which can help to enhance the generalization of the global model. Without ResAtnM, it is noticed that the accuracy is 0.948, which is a reduction from the accuracy with the ResAtnM. This means that feature refinement for attention is an important aspect of extracting defect-relevant structural dependencies in software graphs. Loss of fine-grained node importance information since no residual attention, and over-smoothing in deep graph propagation, reduces discriminative capability. Furthermore, if HSBOA is replaced with standard hyperparameter tuning methods like grid search and random search, the performance would also be decreased. The performance degradation of grid search provides accuracy of 0.951 and the random search is below that level with accuracy of 0.944. This shows that HSBOA can achieve better adaptive optimization by optimizing the parameters of attention weight and model configuration, and balancing exploration and exploitation well. The ablation results demonstrate that each component plays a significant role in the overall performance, with FedRWAvg and the ResAtnM performing the most impactful. The effectiveness of the proposed framework is seen consistently across all ablation variants, which demonstrates that the improvements are deliberate, and not merely due to the combination of federated learning, graph representation learning, and bio-inspired optimization strategies.

Table 8. Performance of ablation experiment

Configuration	Accuracy	Precision	Recall	F1-Score	FPR
Full Model (Proposed)	**0.973**	**0.972**	**0.972**	**0.972**	**0.015**
W/o FedRWAvg (FedAvg used)	0.956	0.954	0.955	0.954	0.032
W/o ResAtnM (GNN only)	0.948	0.946	0.947	0.946	0.039
W/o HSBOA (Grid Search used)	0.951	0.949	0.95	0.949	0.036
W/o HSBOA (Random Search used)	0.944	0.942	0.943	0.942	0.043

5.3. Statistical validation analysis

The results shown in Table 9 illustrate the statistical strength and reliability of the proposed framework under heterogeneous cross-project defect prediction scenarios. All experiments were repeated independently 10 times, with different seeds for the random initialization, and the resulting data are presented as mean ± standard deviation results, which are more realistic than those obtained in the case of a single-run evaluation. The FedHCPDP model had the highest accuracy of 0.973±0.004, the highest F1-score of 0.972±0.004 and the lowest variance among all other competing models. The small standard deviation verifies the stability of the predictions obtained by the proposed federated SOMRAGNN architecture, which also yields reproducible predictions when repeated runs of the algorithm are executed. The Wilcoxon signed-rank test was also performed as a statistical significance analysis to further support the superiority of the proposed framework. All of the comparison methods provide the p-values less than 0.05, which suggest that the improvements in performance that are realized with FedHCPDP are statistically significant and not attributable to random variation. This result shows that the ranking-guided federated aggregation, residual attention graph learning, and HSBOA optimization all play a significant role in prediction performance. Additionally, to avoid the leakage of data in the federated network and overly favorable evaluation conditions, all data preprocessing and SMOTE oversampling steps were confined to the respective training subsets of each federated client. There is no data leakage between validation and test data sets during model training and optimization. The leakage-aware evaluation protocol helps guarantee that the results reflect how the CPDP performs in the real world and not from hidden leakage of information or improper pre-processing.

Table 9. Performance of statistical validation results on AEEEM Dataset

Model	Accuracy (Mean ± Std)	Precision (Mean ± Std)	Recall (Mean ± Std)	F1-Score (Mean ± Std)	p-value
TriStage-CPDP	0.921 ± 0.011	0.918 ± 0.013	0.919 ± 0.012	0.918 ± 0.011	0.0031
ECFML	0.936 ± 0.009	0.934 ± 0.010	0.935 ± 0.009	0.934 ± 0.010	0.0054
SCAG-LSTM	0.948 ± 0.008	0.946 ± 0.009	0.947 ± 0.008	0.946 ± 0.008	0.0082
RH-CPDP	0.956 ± 0.007	0.954 ± 0.007	0.955 ± 0.008	0.954 ± 0.007	0.0115
Proposed	**0.973 ± 0.004**	**0.972 ± 0.004**	**0.972 ± 0.005**	**0.972 ± 0.004**	—

5.4. Discussion

The experimental results of the proposed framework confirm the effectiveness of the proposed framework in heterogeneous cross-project defect prediction situations. For RQ1 (Source Project Relevance), the results show that the proposed ranking-based selection mechanism improves the transfer quality in identifying the most relevant source projects, based on Cosine Similarity, Entropy Difference, and Wasserstein Distance. The benefits of selecting source projects have been observed across all datasets from PROMISE to SOFTLAB, NASA, AEEEM and ReLink are found to be consistent. The result confirms the importance of relevance-aware project selection in heterogeneous CPDP scenarios, where the indiscriminate use of the sources leads to the loss of prediction accuracy. Regarding RQ2 (Structural Representation Learning), the superiority of SOMRAGNN over baseline models, underscores the significance of graph-based modeling for software systems. The multi-scale graph convolution mechanism captures the interactions between modules and the structural dependencies between the modules effectively, and the residual attention component further improves the learning of the defect-sensitive feature. The findings of the improved F1-score and recall values for all datasets show the substantial improvement in the defect prediction ability when using structural learning, especially in high dimensional dataset like AEEEM. For RQ3 (federated aggregation effectiveness), the proposed FedRWAvg strategy outperforms the other aggregated federated learning methods such as FedAvg in ablation experiment. The results show that using a ranking based relevance weights during aggregation results in better informed construction of a global model. FedRWAvg is more informative and similar source projects to the conventional methods which treat all clients equally; and thus, the model can improve the generalization ability under heterogeneous conditions. In the experimental comparisons of RQ4 (optimization and model adaptability), the results indicate that the proposed optimization strategy of HSBOA improves the stability of convergence and overall prediction performance compared to the conventional grid search and random search methods. By combining the two optimizers, Secretary Bird Optimization and Bobcat Optimization, a hybrid optimization paradigm is obtained that supports the exploration-exploitation trade-off, while offering the flexibility of tuning of SOMRAGNN parameters to different project distributions. Repeat runs show low standard deviation values, which are another indication of increased stability and robustness.

6. FUTURE RESEARCH DIRECTIONS

Although the proposed FedHCPDP framework demonstrates strong performance across multiple heterogeneous datasets, several important research directions remain open for further exploration. A major trend is the incorporation of dynamic software evolution information in the existing static graph-based modeling approach. In a real-world software system, code structure is always evolving as a result of changes, bug fixes, refactoring, etc. The use of temporal dependency graphs or evolving graph neural networks may further help the model capture temporal changes in the behavior that lead to defects. Another important area is to enhance the efficiency of communication in FL settings. The proposed framework requires that the multiple source projects are involved in federated aggregation, which can lead to communication overhead in large-scale industrial systems. Possible future research directions include model

compression methods, gradient quantization or reducing the amount of communication to make the model more effective and predictive without increasing bandwidth. Another key open challenge with CPDP systems is explainability. The proposed SOMRAGNN model has high predictive performance, but the information in the decision-making process is still partially opaque. Explainable AI research can be applied to future studies to determine which software modules, dependencies, or metrics most strongly associate with predictions, and improve trust and uptake in industrial environments. Another pathway is continuous learning in federated CPDP environments. Software projects are often modified; new projects are added over time. To improve the scalability and applicability of the system, incremental federated learning approaches that can adapt to new data without retraining from the beginning would be beneficial. Furthermore, the existing framework is binary classification of defects. The model can be extended to predict or localize multiple severity levels in the case of defects in software engineering teams for more practical insights. This would allow critical defects to be prioritized and resources more efficiently allocated when maintaining software. Another constraint is the cross-language and cross domain applicability. The majority of the available datasets, such as those used in this work, are based on the Java programming language or similar languages. The proposed framework would benefit from further research to evaluate its performance when applied to the generalized cases of using different programming languages, frameworks, and development environments. Lastly, FL improves data privacy, but further security measures like secure multi-party computation, differential privacy, or blockchain-based model verification can be incorporated to further bolster information security. These improvements will help to better support sensitive environments for industry and enterprise level software ecosystems.

7. CONCLUSION

The study proposed FedHCPDP to address the challenge of reliable software defect prediction in heterogeneous software repositories. The framework integrates ranking-based source project selection, SOMRAGNN, HHSBOA, and FedRWAvg within a unified federated learning architecture. The proposed ranking mechanism is able to identify the most relevant source projects by Cosine Similarity, Entropy Difference and Wasserstein Distance, which reduces negative transfer effects as the knowledge is transferred. The proposed SOMRAGNN model can effectively learn complex software module relationships by combining multi-scale graph convolution and residual attention learning, with the ability to extract local and global structural information and improve the problem of over-smoothing in deep graph networks. Moreover, the proposed hyperparameter optimization algorithm of HSBOA has a significant improvement to the adaptive hyperparameter optimization in the process of optimization, which can achieve the balance between global exploration and local exploitation. The FedRWAvg aggregation strategy is also useful for improving the performance of federated learning by giving more weight to the contribution of source projects with high relevance to the global model construction. The proposed FedHCPDP framework was evaluated on various datasets such as PROMISE, SOFTLAB, NASA, AEEEM, and ReLink, with the result of the proposed framework consistently outperforming baseline methods. The proposed FedHCPDP framework is a scalable, privacy-preserving solution that is both heterogeneity-adaptive and software defect predication. The framework can be further extended in the future, with the addition of dynamic software metrics, explainable graph learning mechanisms, transformer-based architectures and communication-efficient federated optimization strategies for large-scale industrial software systems.

REFERENCES:

[1] Malik, M.A.I., Carver, J.C. and Eisty, N.U., 2026. Peer code review in research software development: The research software engineer perspective. Empirical Software Engineering, 31(2), p.42.

[2] Retouniotis, A., Papadopoulos, Y. and Sorokos, I., 2025. Andromeda: A model-connected framework for safety assessment and assurance. Journal of Systems and Software, 220, p.112256.

[3] Nevendra, M. and Singh, P., 2022. A survey of software defect prediction based on deep learning. *Archives of Computational Methods in Engineering*, *29*(7), pp.5723-5748.

[4] Parashar, A., Kumar Goyal, R., Kaushal, S. and Kumar Sahana, S., 2022. Machine learning approach for software defect prediction using multi-core parallel computing. Automated Software Engineering, 29(2), p.44.

[5] Jiang, S., Chen, Y., He, Z., Shang, Y. and Ma, L., 2024. Cross-project defect prediction via semantic and syntactic encoding. Empirical Software Engineering, 29(4), p.80.

[6] Ge, S., Qin, F., Wan, X., Liu, Y., Dai, Q. and Zheng, Z., 2026. ARFT-Transformer: Modeling Metric Dependencies for Cross-Project Aging-Related Bug Prediction. Journal of Systems and Software, p.112795.

[7] Nikravesh, N. and Keyvanpour, M.R., 2026. MCOT-KB: Multi-source cross-project defect prediction with optimal transport domain adaptation and KMMBagging. Software Quality Journal, 34(1), p.6.

[8] Saeed, M.S. and Saleem, M., 2023. Cross project software defect prediction using machine learning: a review. *International Journal of Computational and Innovative Sciences*, *2*(3), pp.35-52.

[9] Arumugam, V.K., Reddy, A.A.K., Anand, C., Pabi, D.A., Perumal, C. and Madhan, A., 2025, September. IoT-based Renewable Energy Integration and Optimization in Smart Power Grids. In 2025 6th International Conference on Electronics and Sustainable Communication Systems (ICESC) (pp. 105-110). IEEE.

[10] Kwon, S., Ryu, D. and Baik, J., 2023. An effective approach to improve the performance of eCPDP (early cross-project defect prediction) via data-transformation and parameter optimization. Software Quality Journal, 31(4), pp.1009-1044.

[11] Li, Z., Niu, J. and Jing, X.Y., 2024. Software defect prediction: future directions and challenges. Automated Software Engineering, 31(1), p.19.

[12] Qiu, S., Huang, H., Kuang, Y., Luo, H. and Liu, X., 2025. Enhancing line-level defect prediction using bilinear attention fusion and ranking optimization. Empirical Software Engineering, 30(5), p.116.

[13] Song, H., Pan, Y., Guo, F., Zhang, X., Ma, L. and Jiang, S., 2024. ConCPDP: A Cross-Project Defect Prediction Method Integrating Contrastive Pretraining and Category Boundary Adjustment. *IET Software*, *2024*(1), p.5102699.

[14] Omondiagbe, O.P., Licorish, S.A. and MacDonell, S.G., 2024. Improving transfer learning for software cross-project defect prediction. *Applied Intelligence*, *54*(7), pp.5593-5616.

[15] Song, H., Pan, Y., Guo, F., Zhang, X., Ma, L. and Jiang, S., 2024. ConCPDP: A Cross-Project Defect Prediction Method Integrating Contrastive Pretraining and Category Boundary Adjustment. *IET Software*, *2024*(1), p.5102699.

[16] Zhang, Z., Luo, S., Liu, L. and Pan, L., 2026. Ache-Fuzz: Constraint-Aware Fuzzing for Vulnerability Discovery in Distributed Deep Learning Frameworks. Journal of Systems and Software, p.112796.

[17] Arumugam, V.K., Gowri, A., Ravichandran, S.S., Kushwah, D.S., Vengaimarbhan, D. and Chandrasekar, T., 2025, October. CapsNet–CNN: AI-Powered Voice Authentication System for Secure Online Banking Transactions. In 2025 2nd International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF) (pp. 1-8). IEEE.

[18] Abadeh, M.N., 2024. Knowledge-enhanced software refinement: leveraging reinforcement learning for search-based quality engineering. Automated Software Engineering, 31(2), p.57.

[19] Zhong, Z., Li, C.T. and Pang, J., 2023. Hierarchical message-passing graph neural networks. *Data Mining and Knowledge Discovery*, *37*(1), pp.381-408.

[20] Vashisht, R. and Rizvi, S.A.M., 2023. Feature engineering to heterogeneous cross software projects defect prediction: a Novel framework. *Arabian Journal for Science and Engineering*, *48*(2), pp.2539-2560.

[21] Saeed, M.S. and Saleem, M., 2023. Cross project software defect prediction using machine learning: a review. *International Journal of Computational and Innovative Sciences*, *2*(3), pp.35-52.

[22] Zhou, Y., Cui, F., Che, J., Ni, M., Zhang, Z. and Li, J., 2025. Elastic Balancing of Communication Efficiency and Performance in Federated Learning with Staged Clustering. *Electronics*, *14*(4), p.745.

[23] Bal, P.R. and Kumar, S., 2025. Cross Project Defect Prediction using Dropout Regularized Deep Learning and Unique Matched Metrics. *ACM Transactions on Management Information Systems*, *16*(3), pp.1-32.

[24] Zhang, N., Zhu, K. and Zhu, D., 2024. IAPCP: An Effective Cross-Project Defect Prediction Model via Intra-Domain Alignment and Programming-Based Distribution Adaptation. *IET Software*, *2024*(1), p.5358773.

[25] Abdu, A., Zhai, Z., Abdo, H.A., Lee, S., Al-masni, M.A., Gu, Y.H. and Algabri, R., 2025. Cross-project software defect prediction based on the reduction and hybridization of software metrics. *Alexandria Engineering Journal*, *112*, pp.161-176.

[26] Zou, Y. and Wang, H., 2025. A three-stage cross-project defect prediction framework based on feature representation and knowledge transfer. *Complex & Intelligent Systems*, *11*(11), pp.1-21.

[27] Tao, H., Fu, L., Cao, Q., Niu, X., Chen, H., Shang, S. and Xian, Y., 2024. Cross-Project Defect Prediction Using Transfer Learning with Long Short-Term Memory Networks. IET Software, 2024(1), p.5550801.
[28] Setiawan, B. and Subekti, A., 2025. Multi View Natural Network for Cross-Project Software Defect Prediction. *INSYST: Journal of Intelligent System and Computation*, *7*(1), pp.41-53.
[29] Javed, K., Shengbing, R., Asim, M. and Wani, M.A., 2024. Cross-Project defect prediction based on domain adaptation and LSTM optimization. *Algorithms*, *17*(5), p.175.
[30] Wang, X., Lu, L., Tian, Q. and Lin, H., 2024. IC-GraF: An Improved Clustering with Graph-Embedding-Based Features for Software Defect Prediction. *IET Software*, *2024*(1), p.8027037.
[31] Saraireh, J., Agoyi, M. and Kassaymeh, S., 2025. Adaptive ensemble learning model-based binary white shark optimizer for software defect classification. *International Journal of Computational Intelligence Systems*, *18*(1), p.14.
[32] Chen, H., Yang, L. and Wang, A., 2024. Efficient cross-project software defect prediction based on federated meta-learning. *Electronics*, *13*(6), p.1105.
[33] PROMISE Dataset. NASA promise dataset repository. Available from: https://github.com/jalaxy33/PROMISE-dataset. Accessed November 2025.
[34] Chen, H., Yang, L. and Wang, A., 2024. Efficient cross-project software defect prediction based on federated meta-learning. Electronics, 13(6), p.1105.
[35] NASA Dataset: https://github.com/ApoorvaKrisna/NASA-promise-dataset-repository, Accessed on November 2025.
[36] AEEM Dataset. Available from: https://figshare.com/articles/dataset/Dataset_For_Software_Defect_Predictions/19516858. Accessed November 2025.
[37] ReLink Dataset. Available from: http://cse.hkust.edu.hk/~scc/ReLink.htm. Accessed November 2025.