Journal of Electronic Research and Application, 2026, Volume 10, Issue 6
http://ojs.bbwpublisher.com/index.php/JERA
Online ISSN: 2208-3510
Print ISSN: 2208-3502

Research on Multi-feature Fusion Method for Software Defect Prediction Based on Industrial Internet of Things

Jinpeng Xu

The Third Research Institute of the Ministry of Public Security, Shanghai 200031, China

Abstract: With the rapid development of Industrial Internet of Things technology, the scale of its software is expanding, the complexity of the system continues to rise, and the problem of software defects has become increasingly prominent. Software defect prediction technology can locate potential defects in advance and improve software reliability. However, most of the traditional software defect prediction methods rely on a single code metric feature, which makes it difficult to fully characterize the complex characteristics of Industrial Internet of Things software in the semantic information, program structure, and software evolution process, resulting in limited prediction performance. In view of the above problems, this paper focuses on the research of multi-feature fusion in software defect prediction in Industrial Internet of Things scenarios, focusing on the analysis of the role of different types of features in defect prediction and their fusion mechanism. Firstly, the features of code metrics, semantic features, and structure features involved in Industrial Internet of Things software defect prediction are analyzed. Secondly, the influence of different feature fusion methods on prediction performance is studied, including feature concatenation, weighted feature concatenation, attention fusion, and gating fusion. The research results have a certain reference value for Industrial Internet of Things software quality assurance and intelligent defect analysis.

Keywords: Industrial internet of things; Software defect prediction; Multi-feature fusion

Online publication: Jul 23, 2026

1. Introduction

Software defect prediction is a method to predict potential defect modules by analyzing historical software data, which can help developers improve the efficiency of software testing and repair, thereby reducing the cost of software maintenance. Traditional software defect prediction methods mainly rely on static code metrics such as lines of code, cyclomatic complexity, and coupling degree, and use machine learning models such as random forest and support vector machine to classify and predict defects [1].

In recent years, with the development of deep learning and program analysis technology, researchers

have begun to pay attention to source code semantic information and program structure information. For example, the abstract syntax tree is used to obtain the syntax characteristics of the code [2]. The control flow graph is used to describe the program execution logic, and the program dependence graph is used to reflect the data dependence relationship [3]. At the same time, a large number of commit records and modification histories generated in the process of software version evolution can also reflect the law of software stability and defect evolution, which can improve the prediction performance of the model [4].

However, relying only on a single code metric feature has limited performance in Industrial Internet of Things defect prediction. Multiple features integrate the advantages of different features, represent the source code in multiple dimensions, and improve the model performance [5,6]. Therefore, this paper focuses on the research of multi-feature fusion for software defect prediction in Industrial Internet of Things scenarios, focusing on the analysis of the characteristics of different types of features and their fusion strategies, and the discussion of the impact of multi-feature fusion on the performance of defect prediction, which provides a reference for Industrial Internet of Things software quality assurance.

2. Multi-feature fusion

2.1. Multi-feature

(1) Traditional feature

Through the static analysis of the software source code, extract its structure, scale, complexity, software dynamic behavior, and other quantitative indicators to form features, focusing on the dynamic data of the software product itself and the development process [7]. Its core logic is: the complexity of the code structure, size, design rationality, and the dynamic behavior in the development process are closely related to the probability of the existence of defects. The more complex the code, the more chaotic the structure, and the more frequent the code changes, the easier it is to hide design defects and logic errors.

(2) Semantic feature

Semantic features mainly analyze the lexical information, context, and program logic meaning in the source code to vectorize the semantic information of the code. Unlike traditional features, which focus on code size and complexity, semantic features have more advantages in identifying similar code fragments [8]. Semantic features usually use deep learning techniques to extract semantic information from the sequence of nodes in the abstract syntax tree. Semantic features can effectively mine the hidden defect information, which is difficult to characterize by traditional metric features, so as to enhance the recognition ability of the model for complex defect patterns.

(3) Structural feature

The overall structural behavior of the software program is modeled mainly by analyzing the control relationship, data dependency, and module call relationship within the program. Software defects are not only related to the content of the code itself, but also closely related to the execution process of the program, the degree of module coupling, and the data transmission path [9]. Complex control branches, deep loop nesting, exception jumps, and highly coupled call structures in programs are often more likely to lead to logic errors and potential defects. By constructing a control flow graph, a program dependence graph, and a function call graph, the interaction between program behavior and modules can be described more comprehensively, which enhances the model's ability to perceive the complexity of program logic and the characteristics of defect propagation.

2.2. Feature fusion method

Because different types of features can describe software characteristics from different perspectives, multi-feature fusion has gradually become an important research direction of software defect prediction. Traditional code metrics features can reflect the complexity and scale information of software, semantic features can describe the functional logic of the program, and structural features can describe the control relationship and dependence relationship of the program. Different features are strongly complementary to each other, and it is often difficult to fully describe the defect patterns in complex software systems only by relying on a single feature. Therefore, it is of great significance to study the fusion method of multiple features for improving the performance of software defect prediction [10].

Currently, feature fusion methods in software defect prediction mainly include feature splicing, feature parameter splicing, attention fusion, and gating fusion, as shown in **Figure 1**.

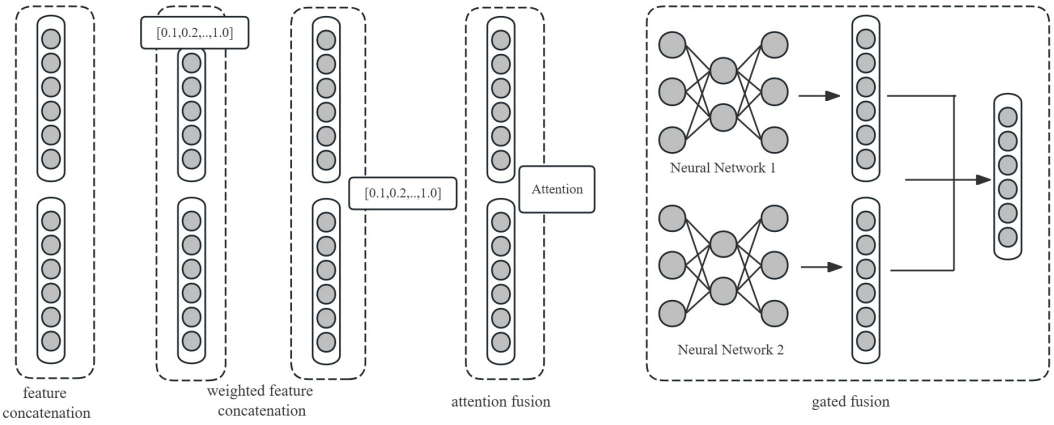

Figure 1. Framework diagram of the multi-feature fusion method.

2.2.1. Feature concatenation

Feature concatenation is a common multi-source feature fusion method, whose core idea is to directly connect feature vectors from different sources according to their dimensions to form a unified high-dimensional feature representation. The feature concatenation method is widely used in traditional machine learning and deep learning models because of its simple implementation, low computational overhead, and easy scalability.

However, this method also has some limitations. Due to the great differences in data distribution, dimension scale, and semantic space of different feature sources, direct stitching can easily lead to feature redundancy and noise accumulation. When the feature dimension is too high, it may also lead to "dimension disaster" and reduce the generalization ability of the model. In addition, by default, different features have the same importance, and the importance of key features can not be learned dynamically, so the feature expression ability is insufficient in complex Industrial Internet of Things software scenarios.

2.2.2. Attention fusion

Attention fusion dynamically learns the importance weights of different features by introducing an attention mechanism, so as to enhance the contribution ability of key features to the prediction results of the model.

Compared with the traditional feature stitching method, attention fusion can adaptively adjust the influence degree of different features according to the feature distribution of the current input sample, so it has a stronger feature expression ability.

Different software modules have different dependencies on various features in the process of defect prediction. For example, for complex control logic modules, structural features may be more important; For code fragments with dangerous function calls, semantic characteristics can be even more critical. Therefore, the importance of different features needs to be dynamically evaluated. Attention fusion can highlight the key defect features, reduce the interference of irrelevant features, and improve the recognition ability of the model for complex defect patterns.

However, there are also some problems with attention fusion. When the training data size is small, the attention mechanism may find it difficult to learn stable weights, resulting in unstable model training. In addition, the attention model has many parameters, and its computational overhead is relatively high in resource-constrained environments such as Industrial Internet of Things edge devices.

2.2.3. Weighted feature concatenation

Weighted feature concatenation is a further improvement based on the traditional feature concatenation method. Its core idea is to introduce the weight adjustment parameter in the feature fusion stage, and give different importance to different types of features or feature dimensions. The weight parameter is usually defined between 0 and 1 to indicate the importance of the corresponding feature to the final fusion representation. The weight can be set according to domain prior knowledge, experimental experience, or model training results, so as to realize the adjustment of different feature contribution degrees.

By assigning different weights or dimensions to different features, weighted feature concatenation can highlight more important feature information for defect prediction and improve the recognition ability of the model for key defect patterns. However, this method still has some limitations. Because the weight parameters are usually adjusted by human experience or experiments, it is difficult to obtain the optimal weight combination automatically. When there are more feature types, the parameter search space will also increase.

2.2.4. Gated fusion

Gated fusion is a dynamic feature fusion method whose core idea is to dynamically control the contribution of different features to the final representation through gating units, so as to achieve more refined feature selection and information fusion.

Unlike the attention mechanism, which mainly emphasizes “which features to pay attention to”, the gating mechanism pays more attention to “how much feature information to retain”. It is essentially similar to the gating structure in recurrent neural networks, which dynamically filters different features by generating gating weights between 0 and 1 through the Sigmoid function.

2.3. Construction of software defect prediction model based on multi-feature fusion

The construction of a software defect prediction model mainly includes data preprocessing, feature extraction, feature fusion, model training, and performance evaluation. Firstly, in the data preprocessing stage, the source code, defect tags, and version data are cleaned and standardized. Subsequently, in the feature extraction stage, multi-source information such as traditional features, semantic features, and structural features is extracted

from the software [11].

After feature extraction, multi-source features are fused by feature splicing, weighted fusion, or an attention mechanism to enhance the model's ability to represent complex defect patterns. Subsequently, the fused features are input into models such as random forest, support vector machine, convolutional neural network, or long and short-term memory network for training, and the mapping relationship between features and defect labels is learned.

Finally, the performance of the model is evaluated by using the test data set, and the commonly used evaluation indicators include Precision, Recall, F1-score, and AUC. Through the comparative analysis of different models and feature fusion methods, the improvement effect of multi-source feature fusion on software defect prediction performance can be further verified.

3. Data set and evaluation index

In the research on Industrial Internet of Things software defect prediction, typical data sets highly consistent with the research theme are selected, as shown in **Table 1**. **Table 1** covers key indicators such as dataset name, source address, and language involved. **Table 2** shows the performance evaluation indicators commonly used in software defect prediction, which provides a clear and convenient reference for relevant researchers.

Table 1. Common data sets

Serial No.	Dataset	Language is involved	Address
1	PROMISE	Java, C, C++	http://promise.site.uottawa.ca/SERepository/datasets-page.html
2	KAMEI	Java, C, C++	https://research.cs.queensu.ca/~kamei/jittse/jit.zip
3	SIR	Java, PHP, C#, C, C++	https://sir.csc.ncsu.edu/portal
4	NASA	Java, PHP, C#, C, C++	https://github.com/klainfo/NASADefectDataset

Table 2. Evaluation index

Serial No.	Indicator name	Value range	Central role
1	Precision	[0,1]	Measure the actual positive proportion of the predicted positive samples, and reflect the accuracy of the prediction results.
2	Recall	[0,1]	Measure the proportion of the actual positive sample that is successfully predicted, reflecting the ability to check all.
3	F1	[0,1]	The harmonic mean of precision and recall is used to comprehensively evaluate the overall classification effect of the model.
4	MCC	[-1,1]	Adapt to the scene of unbalanced positive and negative samples to comprehensively evaluate classification performance.
5	AUC	[0,1]	Evaluate the overall discrimination ability and generalization performance of the model, based on the ROC curve calculation.

4. Concluding remarks

In the Industrial Internet of Things, the construction of a software defect prediction model is an important means to achieve software quality assurance, and multi-source feature fusion can comprehensively depict the potential defect patterns in software from different perspectives. By integrating traditional features, semantic

features, and structural features, the proposed model can not only enhance the ability to represent complex program behaviors but also improve the defect recognition effect of the model in complex Industrial Internet of Things scenarios. With the development of deep learning, graph neural networks, and pre-training models, software defect prediction models will gradually develop in the direction of intelligence, multi-modal, and high robustness, which will provide more effective technical support for the safe and stable operation of Industrial Internet of Things software.

Disclosure statement

The author declares no conflict of interest.

References

[1] Deng T, Deng Y, 2025, Review of Development and Application of Software Defect Prediction Technology in Industrial Internet of Things Environment. Computer Science, 52(S2): 739–749.

[2] Guan X, Zhang H, 2025, Research on Software Defect Prediction Technology Based on Deep Learning. Network Security and Informatization, 2025(6): 56–58.

[3] Phan A, Le Nguyen M, Bui L, 2017, Convolutional Neural Networks over Control Flow Graphs for Software Defect Prediction. Proceedings of the IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), Boston, MA, USA: IEEE, 45–52.

[4] Choudhary G, Kumar S, Kumar K, et al., 2018, Empirical Analysis of Change Metrics for Software Fault Prediction. Computers & Electrical Engineering, 67: 15–24.

[5] Yang M, Yang S, Wong W, 2024, Multi-Objective Software Defect Prediction via Multi-Source Uncertain Information Fusion and Multi-Task Multi-View Learning. IEEE Transactions on Software Engineering, 50(8): 2054–2076.

[6] Siachos I, Kanakaris N, Karacapilidis N, 2025, Software Bug Prediction Using Graph Neural Networks and Graph-Based Text Representations. Expert Systems with Applications, 259: 125290.

[7] Xu J, Guo X, Wang R, et al., 2023, Aggregation Model for Software Defect Prediction Based on Data Enhancement by GAN. Computer Science, 50(12): 24–31.

[8] Shen J, Liu N, Sun H, et al., 2025, Lightweight Semantic Feature Extraction Model with Direction Awareness for Aerial Traffic Object Detection. IEEE Transactions on Intelligent Transportation Systems, 27(4): 4569–4586.

[9] Liu H, Li Z, Zhang H, et al., 2024, CFG2AT: Control Flow Graph and Graph Attention Network-Based Software Defect Prediction. IEEE Transactions on Reliability, 74(3): 3412–3426.

[10] Qiu S, E B, He J, et al., 2025, Survey of Software Defect Prediction Features. Neural Computing and Applications, 37(4): 2113–2144.

[11] Li J, Zhu Y, Yu Q, et al., 2025, Software Defect Prediction Based on Gated Fusion of Pre-Trained Models. Proceedings of the 25th International Conference on Software Quality, Reliability, and Security Companion (QRS-C), IEEE, 204–213.