JIT-Coka: An Improved Framework for Just-in-Time Defect Prediction and Localization Using Fused Features of Code Change

Presentation: Yuguo Liang

Supervisor: Guisheng Fan

CONTENTS

01

Introduction

Introduction

Software defects, also known as bugs or faults, can manifest in various forms such as API misuse, coding errors, style violations, and security vulnerabilities.

Just-**i**n-**T**ime **D**efect **P**rediction and **L**ocalization (**JIT-DP** and **DL**) play a crucial role in software quality assurance by identifying defective code changes and locating faulty lines at the time of code submission.

Fixing Commit Message

Revert "Make VisibleRefFilter.Filter reuse the refs passed from JGit."

This reverts commit b032a529f83892dfbdfb375c47a90d89756dd8ab. This commit introduced an issue where tags were not replicated under certain circumstances.

Bug: Issue 2500
Bug: Issue 1748
Change-Id: I9c902b99c7f656c7002cf3eab9e525f22a22fb85

Defective Commit: b032a529f83892dfbdfb375c47a90d89756dd8ab

2 gerrit-server/src/main/java/com/google/gerrit/server/git/VisibleRefFilter.java

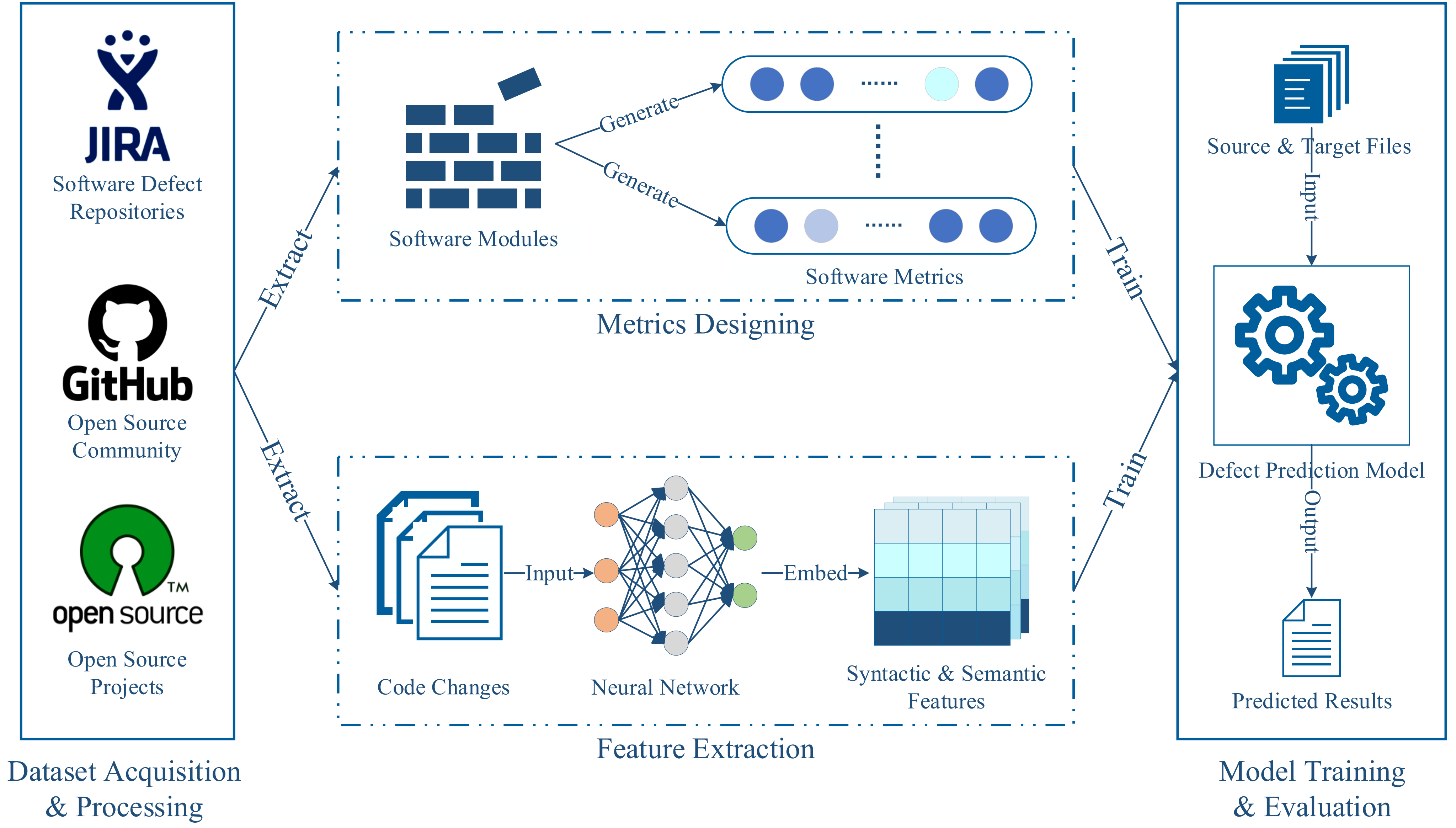
```
    if (!deferredTags.isEmpty() && (!result.isEmpty() || filterTagsSeperately)) {
      TagMatcher tags = tagCache.get(projectName).matcher(
          tagCache,
          db,
-         filterTagsSeperately ? filter(db.getAllRefs()).values() : result.values());
+         filterTagsSeperately ? filter(refs).values() : result.values());
      for (Ref tag : deferredTags) {
```

An example of submission message for defect fix and corresponding defective code changes

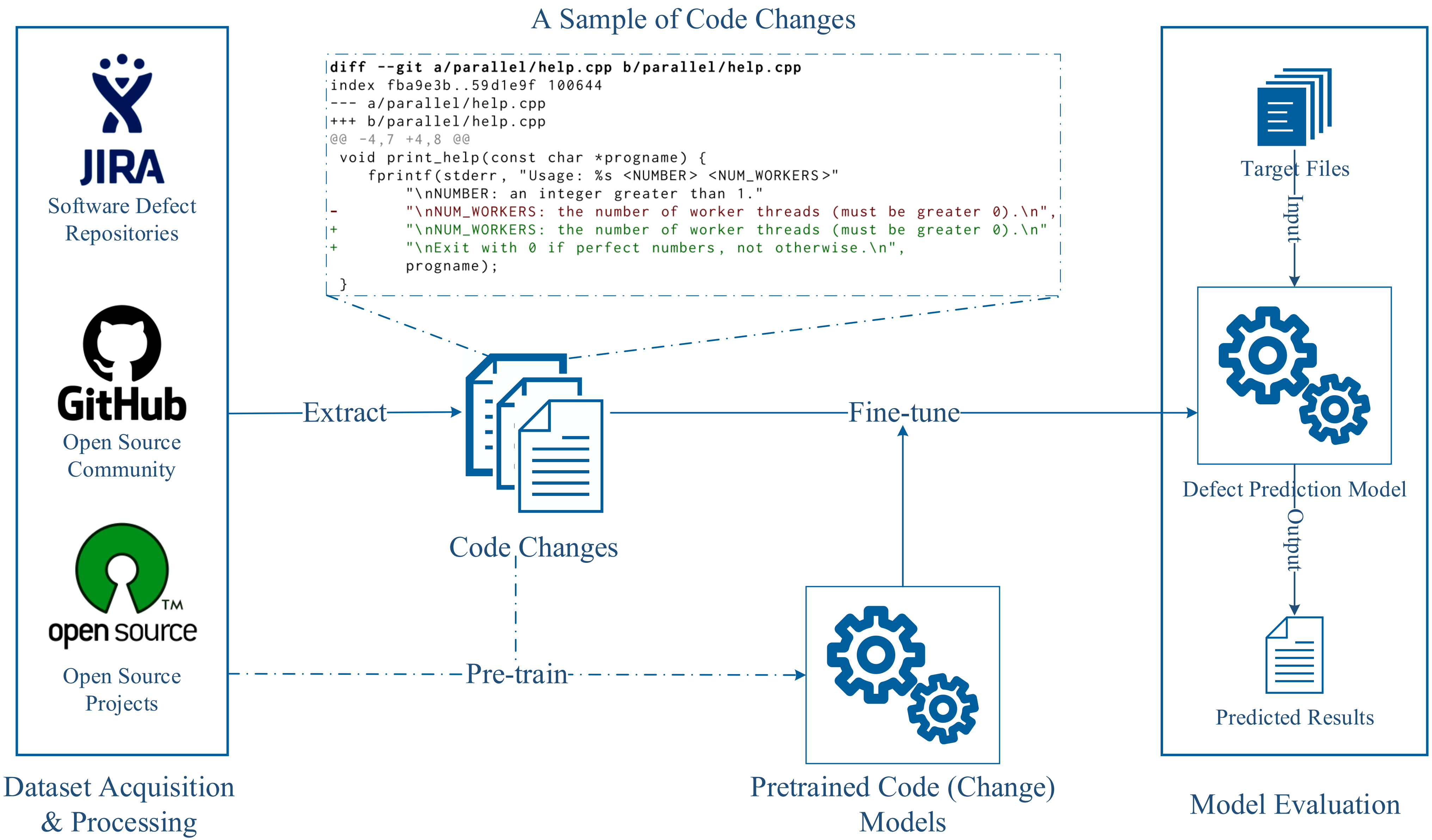
02

Context & Motivation

Status Quo: Machine Learning -> Deep Learning



Status Quo: Deep Learning -> Pre-Trained Models



Motivation

- State-of-the-art (SOTA) JIT-DP and DL approaches such as JIT-Smart that combine expert features with semantic features do not implement mechanisms to explicitly distinguish them.

- More recent JIT-DP and DL frameworks are still built upon encoder-only CodePTMs such as CodeBERT, which are pre-trained on a limited range of programming languages and tasks.

- Most papers rely on AUC-ROC and F1 only. While for imbalanced datasets, Matthews Correlation Coefficient (MCC) is a more comprehensive metric, yet often ignored.

03

Approach

Approach

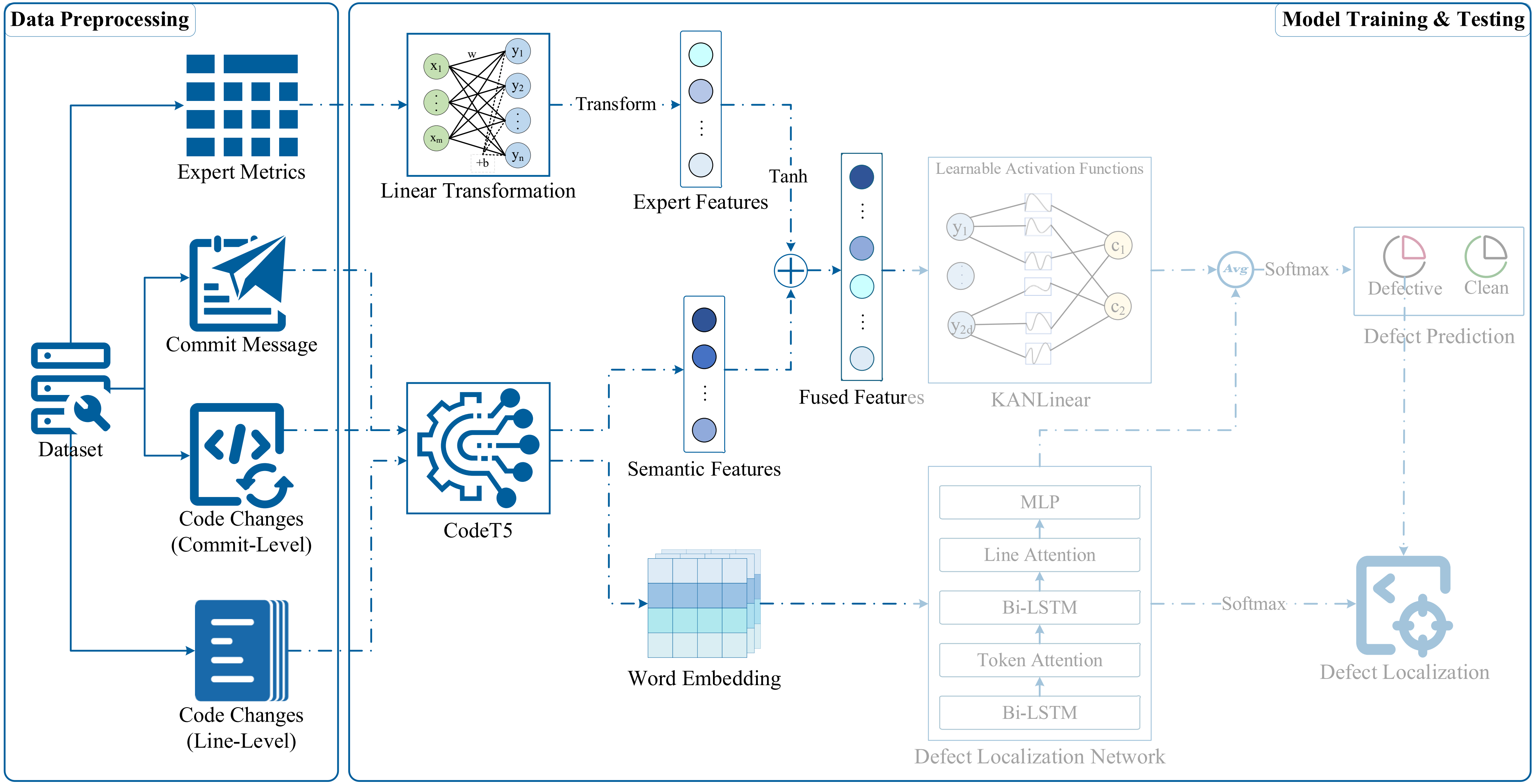


Fig.1：Framework of the JIT-Coka model.

华東理工大學 EAST CHINA UNIVERSITY OF SCIENCE AND TECHNOLOGY

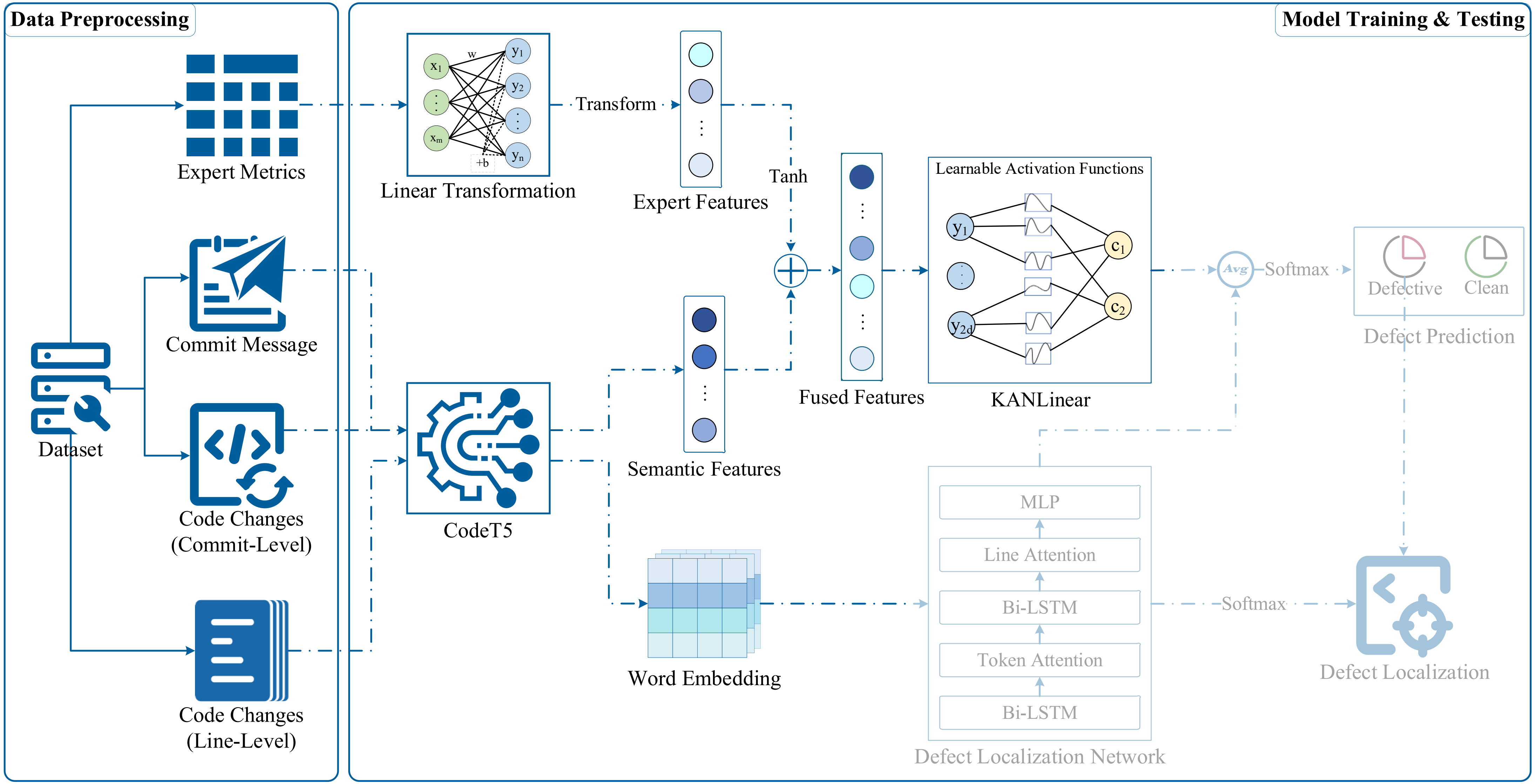


Fig.1：Framework of the JIT-Coka model.

Approach

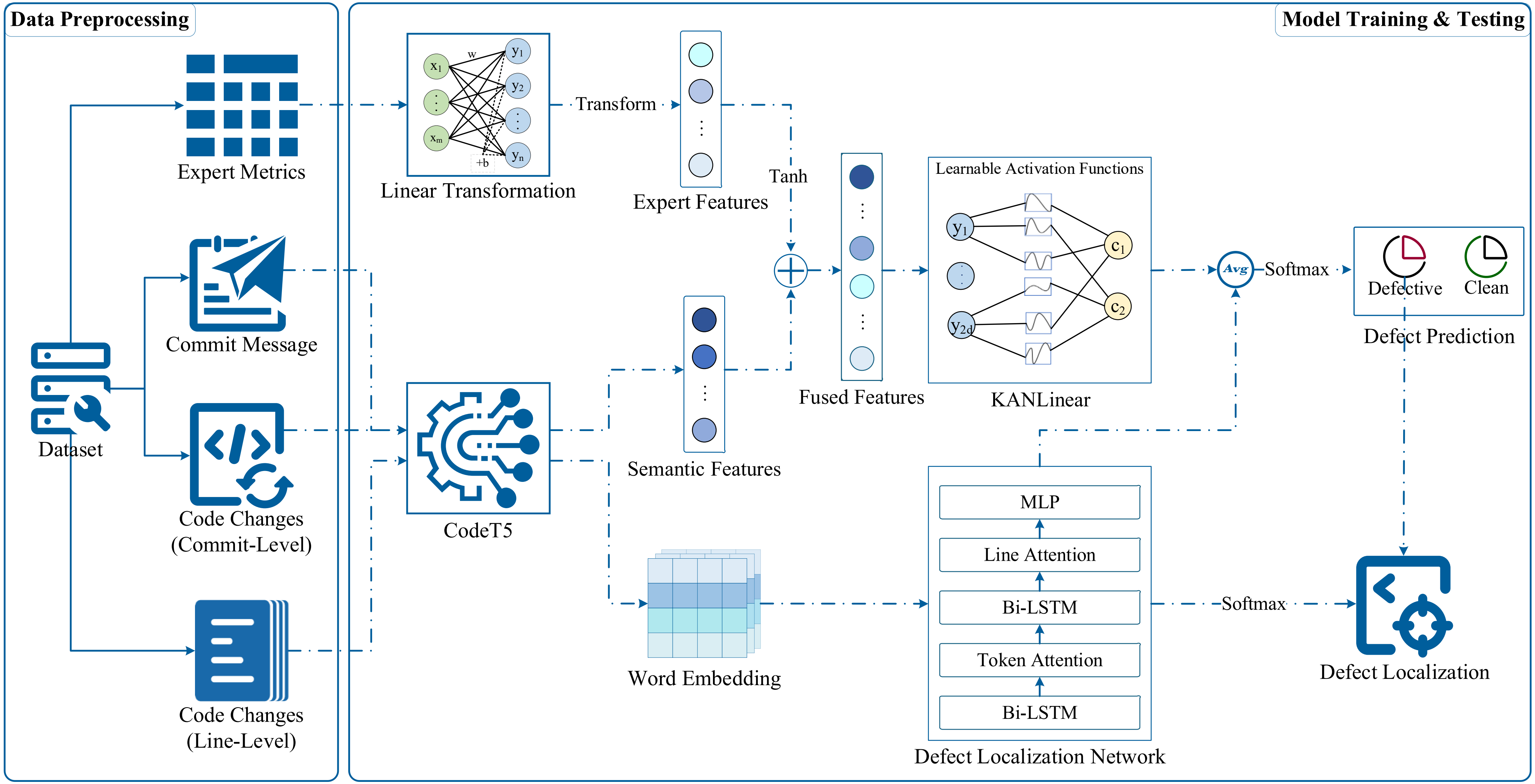


Fig.1: Framework of the JIT-Coka model.

04

Experimental Setup

Experimental Setup-Baselines

華東理工大學
EAST CHINA UNIVERSITY OF SCIENCE AND TECHNOLOGY

DP & DL baselines

4. JITLine (2021)

Random Forest classifier for DP.
LIME model for DL.

5. JIT-Fine (2022)

CodeBERT for feature extraction.
Attention score for DL.

6. JIT-Smart (2024)

CodeBERT for DP.
DLN module for DL.
Introduce Focal Loss.

DP baselines

1. DBN-JIT (2015)

Deep Belief Network

2. DeepJIT (2019)

TextCNN-based model

3. LApredict (2021)

Logistic Regression that uses only one feature.

Experimental Setup-Dataset

Table 1. Statistics of the JIT-Defects4J dataset

Partition	Commit-Level		Line-Level	
	Commit	**Defective (Ratio %)**	**Line**	**Defective (Ratio %)**
Train	16,374	1,390 (8.49%)	117,818	7,872 (6.68%)
Valid	5,465	467 (8.55%)	32,642	2,611 (8.00%)
Test	5,480	475 (8.67%)	26,665	2,111 (7.92%)
Total	27,319	2,332 (8.54%)	177,125	12,594 (7.11%)

Source: Chao Ni, Wei Wang, Kaiwen Yang, Xin Xia, Kui Liu, and David Lo. 2022. *The best of both worlds: Integrating semantic features with expert features for defect prediction and localization*. ESEC/FSE 2022. Association for Computing Machinery, New York, NY, USA, 672–683. https://doi.org/10.1145/3540250.3549165

Experimental Setup-Research Questions

RQ1: What is the best performance that JIT-Coka and relevant baselines can achieve on the *JIT-Defects4J* dataset?

RQ2: How do JIT-Coka and baselines perform over multiple runs on the *JIT-Defects4J* dataset?

RQ3: What is the effectiveness of each component in the JIT-Coka model?

Discussion: How do JIT-Coka and baselines perform on dataset comprising multiple programming languages?

Experimental Setup-Evaluation Metrics

Precision: The proportion of true positives among all predicted positives.

Recall/True Positive Rate: The proportion of actual positives that are correctly identified.

F1-Score: The harmonic mean of precision and recall.

Matthews Correlation Coefficient (MCC): A balanced metric suitable for imbalanced datasets.

Area Under the Receiver Operating Characteristic Curve (AUC-ROC): A threshold-independent metric that measures a classifier's performance across all thresholds.

Top-5/10 Accuracy: Evaluate the model's ability to rank truly defective lines near the top of its predictions.

Wilcoxon Rank-Sum test/Mann-Whitney U Test: Assess the statistical significance of performance differences between all evaluated models across multiple runs.

Experimental Setup-Parameter Settings

Feature Extractor: hidden size 768, dropout 0.1.

Classifier: grid size 0.5, spline order 3, and base activation function SiLU for KANLinear.

Loss Function: α 0.25 and γ 2 for focal loss, λ_{DP} 0.3 and λ_{DL} 0.7 for final loss weight.

Training & Evaluation: batch size 8, learning rate 1e-5, max training steps 50, patience 5.

Repeated Experiments: random seeds 42, 88, 1234, 2024, and 2048.

Efficiency: training ~7.5h and inference ~205s (0.037s per sample) for JIT-Coka (Ours);

training ~10.5h and inference ~192s (0.035s per sample) for JIT-Smart (SOTA).

Environment: NVIDIA Tesla P40 GPU (24GB), Intel(R) Xeon(R) Gold 5118 CPU, 100GB RAM, CentOS 7.6.

05

Results & Analysis

Results & Analysis-RQ1

Table 2. Optimal JIT-DP performance of JIT-Coka and related baselines on the JIT-Defects4J dataset

Model	Precision	Recall	F1-score	MCC	AUC-ROC
Deeper	0.1748	0.4295	0.2485	0.1629	0.6772
LApredict	0.4545	0.0316	0.0591	0.1018	0.6938
DeepJIT	0.2126	**0.6632**	0.3219	0.2724	0.7911
JITLine	**0.6391**	0.1789	0.2796	0.3096	0.8087
JIT-Fine	0.4792	0.3874	0.4284	0.3829	0.8777
JIT-Smart	0.5023	0.4611	0.4808	0.4343	**0.8916**
JIT-Coka	0.5463	0.4842	**0.5134**	**0.4713**	0.8887

Table 3. DL performance of JIT-Coka and related baselines when achieving their optimal JIT-DP performance on the JIT-Defects4J dataset

Model	Accuracy	
	Top-5	Top-10
JITLine	0.1339	0.1214
JIT-Fine	0.1749	0.1672
JIT-Smart	0.5409	0.3943
JIT-Coka	**0.5459**	**0.4038**

RQ1: What is the best performance that JIT-Coka and relevant baselines can achieve on the JIT-Defects4J dataset?

Results & Analysis-RQ2

RQ2: How do JIT-Coka and baselines perform over multiple runs on the JIT-Defects4J dataset?

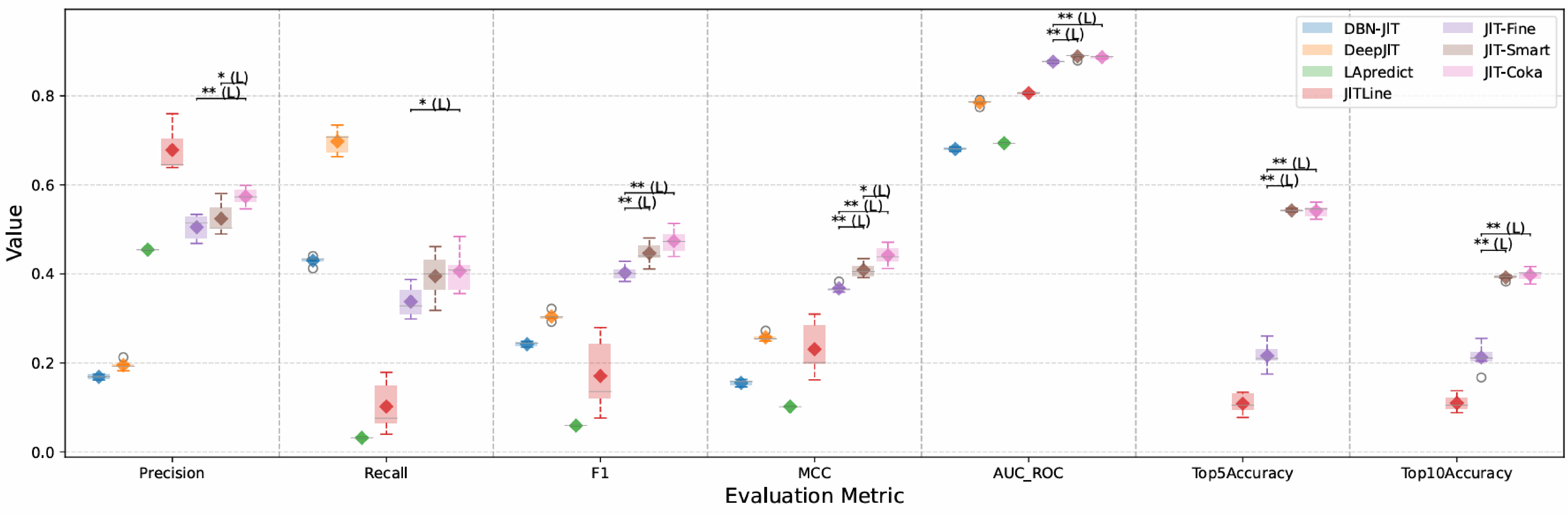


Fig.2: Distributions of JIT-DP and DL performance for JIT-Coka and related baselines on the *JIT-Defects4J* dataset.

Results & Analysis-RQ3

RQ3: What is the effectiveness of each component in the JIT-Coka model?

Table 4. Ablation study of different components in the JIT-Coka model

Model	**JIT-DP Metrics**					**DL Accuracy**	
	Precision	**Recall**	**F1-score**	**MCC**	**AUC-ROC**	**Top-5**	**Top-10**
JIT-Coka	0.5463	**0.4842**	**0.5134**	**0.4713**	0.8887	0.5459	0.4038
– w/o CodeT5	0.5045	0.4674	0.4852	0.4388	**0.8905**	0.5435	**0.4063**
– w/o KANLinear	0.5875	0.3958	0.4730	0.4433	0.8881	**0.5483**	0.4003
– w/o DLN	**0.5893**	0.4168	0.4883	0.4565	0.8894	0.1992	0.2033
– w/o EF	0.3789	0.4547	0.4134	0.3539	0.8610	0.5424	0.3953

Results & Analysis-RQ3

RQ3: What is the effectiveness of each component in the JIT-Coka model?

Table 5. Performance of JIT-Coka with different loss weight ratios for DP and DL

Loss Weight		JIT-DP Metrics					DL Accuracy	
λ_{DP}	λ_{DL}	Precision	Recall	F1-score	MCC	AUC-ROC	Top-5	Top-10
0.1	0.9	0.4837	0.4695	0.4765	0.4277	0.8853	0.5466	0.4028
0.2	0.8	**0.5769**	0.4105	0.4797	0.4467	0.8868	0.5379	0.3973
0.3	0.7	0.5463	**0.4842**	**0.5134**	**0.4713**	**0.8887**	0.5459	0.4038
0.4	0.6	0.5083	0.4526	0.4788	0.4334	0.8882	0.5474	0.4051
0.5	0.5	0.5174	0.4695	0.4923	0.4473	0.8881	0.5478	0.4033
0.6	0.4	0.5208	0.4484	0.4819	0.4382	0.8831	0.5453	0.4069
0.7	0.3	0.4903	0.4779	0.4840	0.4358	0.8885	0.5435	0.4005
0.8	0.2	0.4784	0.4653	0.4717	0.4224	0.8819	**0.5518**	**0.4118**
0.9	0.1	0.4715	0.4695	0.4705	0.4203	0.8863	0.5402	0.4004
1.0	0.0	0.5403	0.4379	0.4837	0.4431	0.8818	0.5328	0.3934

Results & Analysis-Discussion

Discussion: How do JIT-Coka and baselines perform on dataset comprising multiple programming languages?

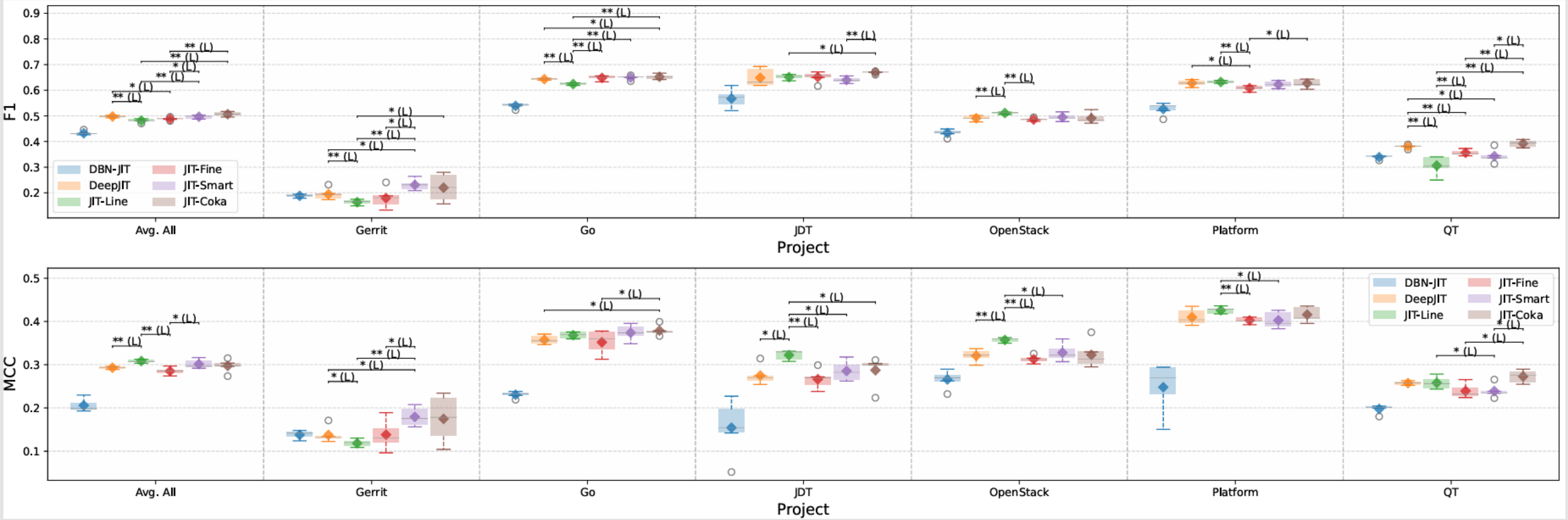


Fig.3: Distributions of JIT-DP performance for JIT-Coka and related baselines on the *LApredict* dataset.

06

Conclusion

Conclusion

GitHub: https://github.com/Hugo-Liang/JIT-Coka
Presentation: https://hugo-liang.github.io/publication/2025-CollaborateCom-JIT-Coka

In this paper, we proposed JIT-Coka, a unified model for Just-in-Time Defect Prediction and Localization, which effectively integrates pre-trained semantic features and handcrafted expert features through an adaptive nonlinear classification module. Leveraging the encoder-decoder architecture of **Co**deT5 and a robust **KA**NLinear classifier, JIT-Coka improves the Precision, F1 and MCC for defect prediction. Moreover, the use of the DLN module enables effective line-level localization, tightly coupled with commit-level classification.

Thanks for listening!

Presentation: Yuguo Liang

Supervisor: Guisheng Fan