What is the cross-domain robustness of LLaMA 3.2 on bug detection when fine-tuned on BugsInPy versus JavaScrip

Assignee Research

June 10, 2026

Abstract

Large language models (LLMs) have demonstrated strong performance on a wide range of software engineering tasks, including code generation and analysis. However, most prior work relies on cloud-based models or specialized hardware, limiting practical applicability in privacy-sensitive or resource-constrained environments. In this paper, we present a systematic empirical evaluation of two locally deployed LLMs, LLaMA 3.2 and Mistral, for real-world Python bug detection using the BugsInPy benchmark. We evaluate 349 bugs across 17 projects using a zero-shot prompting approach at the function leve

1 Introduction

This paper examines: An Empirical Evaluation of Locally Deployed LLMs for Bug Detection in Python Code. Research question: What is the cross-domain robustness of LLaMA 3.2 on bug detection when fine-tuned on BugsInPy versus JavaScript's Defects4J, evaluated using F1-score comparisons across both datasets?.

2 Methodology

Systematic literature search across multiple databases yielded 14 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 4.8/10.

3 Results

14 papers retrieved. 17 claims extracted; 4 independently verified. Quality review score: 4.8/10.

4 Limitations

This report is a machine-generated literature synthesis and does not constitute original research. Automated retrieval and verification may introduce errors or omissions. Review scores reflect automated assessment, not human peer review. Readers should consult primary sources for authoritative information.

5 Extracted Claims

Claim	Verified	Confidence
Locally executed models achieve accuracy between 43% and 45% in bug detection.	✓	0.26
Locally executed models produce a large proportion of partially correct responses that identify problematic code regions	✓	0.32
Performance varies significantly across projects, highlighting the importance of codebase characteristics.	✓	0.23
Local models can identify a meaningful share of bugs, though precise localization remains difficult for locally executed	✓	0.34
Performance degrades when bugs require cross-function reasoning.	×	0.04
Defect complexity is the primary factor governing detection accuracy.	×	0.03
Model performance drops systematically as the number of co-occurring defects increases.	×	0.03
Open-weight models can approach proprietary system performance on converting unstructured bug reports into structured fo	×	0.06
The availability of open-weight models such as LLaMA and Mistral has made local deployment a practical option.	×	0.07
Prior work has largely evaluated large cloud-hosted models.	×	0.07
Codex, a GPT-based model fine-tuned on publicly available code, evaluates its ability to generate functional Python prog	×	0.07
CodeBERT, trained jointly on natural language and programming language data, enables tasks such as code search and docum	×	0.06
AutoFL uses LLMs for fault localization while generating natural language explanations alongside predictions, improving	×	0.04
BugsInPy is a curated set of real Python bugs with reproducible test cases drawn from well-known open-source projects.	×	0.12
Reproducibility issues were identified in the BugsInPy dataset and revisions were proposed to support more reliable eval	×	0.03
Evaluation was extended to multi-vulnerability settings, showing that model performance drops systematically as the numb	×	0.02
A systematic review of LLM-based automated program repair categorized methods and identified open challenges.	×	0.04

References

- http://arxiv.org/abs/2508.11281v3
- http://arxiv.org/abs/2507.21954v2
- http://arxiv.org/abs/2604.23361v1